UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Realizzazione e validazione di un modulo di link management per reti wireless mesh

Relatori:

Prof. Luciano Lenzini

Prof. Enzo Mingozzi

Ing. Luca Bisti

Candidato:

Angelo Colucci

ANNO ACCADEMICO 2008-2009

Sommario

In questo lavoro di tesi viene illustrata la realizzazione di un modulo di link management per reti wireless mesh consistente con le specifiche del draft 3.0 sull' 802.11s. Il modulo è stato sviluppato utilizzando il framework fornito dal software Click Modular Router ed è stato installato su un prototipo di rete mesh 802.11, composto da 5 router equipaggiati ognuno con 2 schede di rete wireless 802.11a/b/g e 2 schede di rete Ethernet.

Dal punto di vista software i nodi sono equipaggiati con un meccanismo di instradamento basato sul label switching conforme allo standard RFC 3031, con un protocollo di distribuzione delle label che consente di installare automaticamente i Label Switched Path (LSP), in accordo all' RFC 5036, e in più fornisce un meccanismo dedicato per l' installazione di rotte di backup (detour), attivabili grazie all' interfaccia FrrDB.

In una seconda fase il modulo di link management è stato esteso con un meccanismo di valutazione della qualità dei link wireless che, interfacciandosi all' FrrDB, comanda l' attivazione dei detour; sono stati quindi effettuati dei test per validare le soluzioni proposte.

INDICE

ELENCO DELLE TABELLE

ELENCO DELLE FIGURE

1 INTRODUZIONE

1.1 MOTIVAZIONI

Le reti wireless 802.11 hanno rappresentato una valida alternativa al dispiegamento di reti cablate nell' ambito della realizzazione di reti informatiche locali (LAN). I fattori che ne hanno determinato il successo sono, da un lato, di natura economica: il costo contenuto dei dispositivi di accesso (AP), con cui viene fornita la copertura di rete, ed il risparmio economico legato ai costi di cablaggio; dall' altro, di natura pratica: la facilità di dispiegamento della rete rende preferibile tale soluzione in tutti quei casi in cui il cablaggio risulterebbe difficile da realizzare.
D' altra parte, negli ultimi anni, le Wireless LAN (WLAN) hanno evidenziato dei limiti in merito ad un loro utilizzo su larga scala. Per connettere più WLAN tra loro, tutti gli AP devono essere collegati attraverso una rete fissa, che funge da backbone per le varie WLAN. Un problema simile si presenta volendo fornire connettività ad Internet ad un certo numero di WLAN; ciascun AP deve essere collegato mediante cavo alla rete di accesso ad Internet.

L' alternativa offerta dalle Mobile Ad Hoc Network (MANET) è scalabile dal punto di vista della copertura geografica. Essa, infatti, è costituita da un insieme di nodi mobili che dinamicamente si autoconfigurano a formare una rete wireless, senza la necessità di un' infrastruttura di telecomunicazione preesistente; ciò è reso possibile dal fatto che ciascun nodo, oltre a comunicare direttamente con i nodi che sono nella propria area di copertura, utilizza i nodi intermedi come router verso le destinazioni al di fuori del proprio range di trasmissione. Rispetto al caso delle WLAN, le MANET definiscono sì un' architettura di rete decentralizzata, facilmente estendibile e adattabile ai cambiamenti della rete, ma isolata, non accessibile ad altro tipo di

reti, per esempio Internet; questo perché le MANET sono state progettate in ambienti militari e utilizzate per applicazioni specifiche (di tipo militare o di protezione civile). Questo spiega la scarsa diffusione commerciale che hanno avuto.

In tale contesto, le Wireless Mesh Networks (WMN), si pongono a metà strada tra le MANET e le WLAN; dalle prime ereditano la capacità di autoconfigurazione e di autoriparazione, sfruttandola per il dispiegamento di una backbone di tipo wireless che colleghi gli AP tra loro; dalle seconde ereditano la possibilità di proporsi come estensione di una infrastruttura di rete preesistente.

Affinché sia possibile definire una backbone di tipo wireless, che garantisca continuità di servizio (caratteristica imprescindibile per una backbone), le WMN devono essere equipaggiate con un meccanismo di forwarding efficiente che minimizzi i tempi di calcolo richiesti per effettuare l' instradamento dei pacchetti, con protocolli di routing capaci di reagire rapidamente ai cambiamenti della topologia, dovuti più spesso alle interferenze sul canale che al guasto di un nodo, con tecniche che sfruttino la ridondanza dei percorsi di rete e, attivando delle rotte di backup prestabilite, garantiscano continuità di servizio, qualora un nodo non sia più raggiungibile. In tal senso nei due lavori di tesi che hanno preceduto questo sono stati implementati un meccanismo di forwarding basato sul label switching, [1], un protocollo di distribuzione delle label che rendesse dinamica l' installazione dei Label Switched Path (LSP) sui nodi e un' interfaccia per l' attivazione delle rotte di backup, attraverso un servizio di fast rerouting [30].

Il fast rerouting è un meccanismo sviluppato nell' ambito delle reti fisse e attivato dal modulo di link management. Nelle reti fisse la fase di link management, che si occupa di determinare la presenza o meno di un link, è facile da realizzare. Infatti, non appena un link si interrompe (nodo-vicino assente o link guasto) il driver che gestisce la scheda di rete notifica l' evento al software di livello data link permettendo così l' attivazione della rotta di backup. Nelle reti wireless se un nodo è assente da ciò non si può dedurre immediatamente che il nodo o il link sia guasto; può accadere infatti che le interferenze sul canale rendano

momentaneamente irraggiungibile il nodo.
Per questo il modulo di link management nelle WMN deve fornire dei meccanismi di analisi della qualità del canale che gli consentano di decidere se questo ultimo possa essere considerato/astratto come un link attivo o inattivo.

1.2 OBIETTIVI

Lo scopo di questo lavoro è quello di realizzare un protocollo di link management per WMN che, integrato con un sistema di misura della qualità del link, permetta di attivare le rotte di backup, installate dal modulo di distribuzione delle label e messe a disposizione dall' interfaccia offerta dal servizio di fast rerouting. Le finalità del lavoro sono:

1. Realizzare un modulo di link management conforme alle specifiche fornite con il draft D3.0 dello standard sullo 802.11s [29] (attualmente non ancora ratificato).

2. Integrare il modulo di link management con un modulo di valutazione della qualità del link e con quello di fast rerouting.

3. Verificare se ci sono effetti positivi legati all' utilizzo del fast rerouting in una WMN.

Lo sviluppo e la validazione di queste funzionalità saranno svolte in un ambiente reale.

1.3 STRUTTURA DELLA TESI

La tesi è così strutturata: nel capitolo 2 vengono descritte le WMN, partendo dallo studio dell' architettura di rete per poi passare, nel paragrafo 2.2, ad una panoramica sulle famiglie di protocollli di routing esistenti; segue la descrizione di alcune implementazioni reali (2.3) e il capitolo si conclude con un breve excursus degli standard per WMN, definiti per diverse teconologie wireless.

Nel capitolo 3 viene presentato il testbed utilizzato in questo lavoro; iniziando dall' analisi delle caratteristiche hardware (3.1) e software (3.2) dei router, vengono poi studiati i componenti che definiscono l' architettura di routing e di forwarding: il protocollo di routing SRCR (3.3), la metrica ETT (3.4), il meccanismo di label switching (3.5), il protocollo di distribuzione delle label (3.6) ed il meccanismo di fast rerouting (3.7). In seguito vengono descritte nel dettaglio la procedura di discovery e quella di link management secondo le specifiche della bozza 3.0 802.11s.

Nel capitolo 4 vengono analizzate le strutture dati definite secondo le specifiche della bozza D3.0 ed utilizzate dal software di link management alla base di questo lavoro.

Infine, nel capitolo 5, verranno illustrati i risultati dei test di validazione e delle prestazioni condotti sul testbed.

2

PANORAMICA SULLE WIRELESS MESH NETWORK

In questo capitolo verrà fatta un' introduzione alle WMN. Si parte dalla descrizione del modello architetturale di una WMN (2.1). Analizzeremo quindi il funzionamento dei protocolli di routing esistenti secondo le caratteristiche generali relative alla famiglia di appartenenza: proactive, reactive e hybrid (2.2). Verranno illustrati alcuni esempi di testbed realizzati da altre istituzioni universitarie (2.3). Infine nel paragrafo (2.4) descriveremo gli standard aperti definiti per le WMN.

2.1 ARCHITETTURA DI UNA WMN

Una WMN è costituita da un insieme di nodi wireless che, dislocati in una area, si auto-organizzano a formare una rete magliata in cui la comunicazione tra i nodi avviene secondo un modello multihop. Questo è reso possibile grazie all' utilizzo di nodi dedicati (chiamati mesh-router) in grado, oltre che di trasmettere e ricevere dati, di fornire un servizio di trasporto wireless a quei nodi, mesh router o terminali-utente (mesh-client), la cui destinazione non è direttamente raggiungibile perché al di fuori della propria area di copertura.

Oltre al modello di comunicazione multihop, un' altra caratteristica peculiare delle WMN è la capacità di operare sia indipendentemente che come estensione di una infrastruttura di rete esistente. Infatti i mesh router possono essere collegati come mesh-gateway ad una rete fissa, che, ad esempio, può fornire lo accesso ad Internet; in un contesto 802.11 un mesh router può essere configurato su uno o più AP che realizzano le rispettive WLAN, collegandole fra loro. Questa proprietà configura le WMN come una backbone di tipo wireless, in grado di collegare reti diverse tra loro. La figura 1 mostra un' architettura di rete

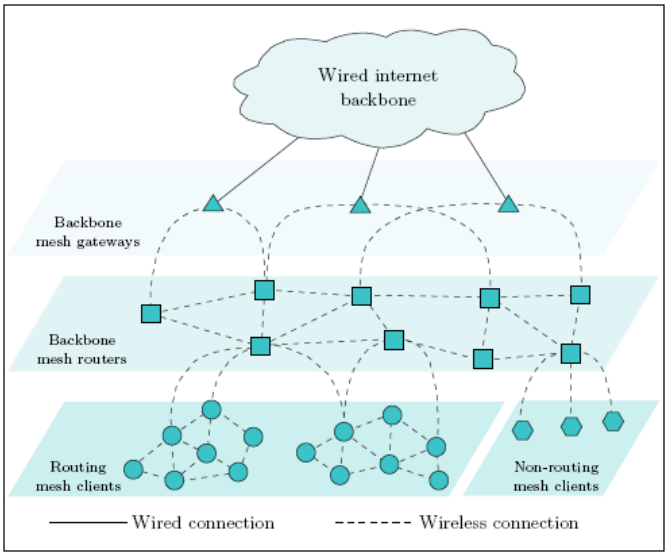

Figura 1: Wireless Mesh Network

gerarchica e a livelli che sintetizza i vari approcci e contesti in cui una WMN viene dispiegata, chiarandone la collocazione e il ruolo svolto.

Al livello più alto dell' architettura ci sono i *backbone mesh gateway* connessi ad Internet mediante cavi di rete. Essi forniscono un accesso wireless ad Internet ai nodi posti al secondo livello, i *backbone mesh router*. Essi formano insieme ai backbone mesh gateway la backbone di tipo wireless. Al livello più basso ci sono i *mesh client*, ossia i terminali utente. I mesh client sono di due tipi: i *routing mesh client* e i *non-routing mesh client*. I primi sono in grado di comunicare tra loro secondo un modello multihop, mentre i secondi sono collegati al mesh router allo stesso modo con cui i client 802.11 sono associati all' AP. Le due tipologie di client comunicano tra loro e accedono ad Internet grazie al servizio di trasporto fornito dalla backbone wireless. Di seguito vengono riassunti i principali vantaggi legati all' utilizzo di una architettura di rete wireless-mesh [2]:

RIDUZIONE DEI COSTI DI ATTIVAZIONE Il problema di fornire connettività ad Internet negli spazi pubblici, come aeroporti e stazioni, fino ad ora è stato risolto con il dispiegamento di AP che forniscono il collegamento ad internet,

tramite la rete fissa a cui sono collegati, e un' area di copertura limitata per l' accesso dei wireless-client. Per poter coprire aree estese bisogna dislocare molti AP, ciascuno dei quali deve essere collegato alla rete fissa. Questa soluzione è costosa, non-scalabile, e richiede tempi lunghi per il dispiegamento. L' utilizzo di una WMN riduce notevolemente i costi legati all' infrastruttura perché ha bisogno di pochi nodi collegati alla rete fissa.

DISPIEGAMENTO SU LARGA SCALA Il paradigma di comunicazione multihop offre la possibilità di effettuare comunicazioni su lunga distanza grazie al servizio di instradamento fornito dai nodi intermedi.

AFFIDABILITÀ La backbone wireless fornisce molteplici percorsi di rete per una stessa coppia di nodi, aumentando così l' affidabilità della comunicazione ed evitando che si presentino single point of failure e colli di bottiglia nella rete.

AUTOGESTIONE I nodi sfruttano le funzionalità mesh per scoprire tutti i possibili vicini e per determinare i percorsi migliori verso la rete fissa. Questo significa che la rete è in grado di reagire e auto-riconfigurarsi alla luce dei cambiamenti topologici che avvengono.

2.2 PROTOCOLLI DI ROUTING

Per dotare le WMN di un' architettura decentralizzata, capace di auto-configurarsi e di rispondere ai cambiamenti topologici della rete è necessario definire dei protocolli di routing dedicati allo scopo. Tre sono le categorie di protocolli emerse: *proactive*, *reactive* e *hybrid*.

Protocolli proactive

In questo tipo di approccio i nodi vengono a conoscenza della intera topologia della rete tramite lo scambio periodico di mes-

saggi contenenti informazioni sulle destinazioni raggiungibili e sui cambiamenti topologici della rete. Ogni nodo deve mantenere una lista di tutte le destinazioni e delle possibili rotte per raggiungerle, che viene aggiornata scambiando periodicamente informazioni sulla intera routing table con gli altri nodi, anche in assenza di traffico dati sulla rete. A questa famiglia di protocolli appartengono l' *Optimized Link State Routing Protocol* (OLSR) e il *Destination Sequenced Distance Vector* (DSDV)[1]. Questo tipo di protocolli hanno lo svantaggio legato alla quantità di informazioni di routing che ciascun nodo deve mantenere/scambiare.

Protocolli reactive

Nei protocolli reactive la rotta verso una certa destinazione viene determinata solo su richiesta di un nodo mittente quando deve trasmettere dei dati ad un nodo destinatario per il quale non conosce il percorso di rete. In tal caso il nodo mittente costruisce un messaggio di Route Request nel quale specifica lo identificatore del nodo destinatario che si vuole raggiungere e lo trasmette in broadcast all' interno della rete. Quando questo messaggio raggiunge il nodo di destinazione o un nodo che è a conoscenza di un percorso verso la destinazione stessa viene inviato un messaggio di Route Reply verso il nodo che ha iniziato questo processo, utilizzando le informazioni salvate sui nodi intermedi al passaggio del messaggio di Route Request. Una volta che il percorso è stabilito il nodo sorgente lo memorizza localmente. Il traffico può iniziare a fluire dalla sorgente verso la destinazione.

Il processo di instradamento può essere effettuato in forma di source routing (la lista dei nodi da attraversare viene specificata dal mittente all' interno dell' intestazione del pacchetto dati) oppure hop-by-hop (ogni pacchetto contiene soltanto l' indirizzo del destinatario e del next-hop) [1]. La soluzione basata sul source routing è efficiente se utilizzata in reti composte da pochi nodi, altrimenti diventa non trascurabile l' overhead introdotto dal trasporto di un elevato numero di indirizzi all' interno dei pacchetti. Tra i protocolli reactive basati sul source routing ricor-

diamo il *Dynamic Source Routing* (DSR), mentre per quelli basati sull' hop-by-hop ricordiamo l' *Ad-hoc On-Demand Distance Vector* (AODV). I principali svantaggi associati all' utilizzo di protocolli di tipo reactive sono:

1. L' overhead legato alla fase di setup di una rotta, eseguita tutte le volte che un nodo mittente non conosce il percorso per trasmettere i dati verso un nodo destinatario.
2. L' utilizzo da parte di ogni nodo mittente di rotte subottimali che non tengono conto della formazione di percorsi migliori verso una stessa destinazione.

Protocolli hybrid

A questa famiglia appartengono quei protocolli di routing che si basano su una combinazione di meccanismi tipici dei protocolli proactive e di quelli reactive. I nodi inizialmente raccolgono informazioni sull' intera topologia della rete, utilizzando le stesse tecniche adottate dai protocolli proactive; in seguito l' aggiornamento delle singole rotte viene fatto con tecniche di query-on-demand tipiche dei protocolli reactive. Esempi di questo tipo di protocolli sono l' *ARPAM*, l' OrderOne Routing Protocol (OORP), l' SRCR [10] e l' *Hybrid Routing Protocol for Large Scale Mobile Ad Hoc Networks with Mobile Backbones* (HRPLS). I principali svantaggi di questi algoritmi sono:

1. La dipendenza dell' efficienza del protocollo dal numero di nodi attivi all' interno della rete.
2. La dipendenza della capacità di risposta ad una elevata domanda di traffico dal gradiente del volume del traffico.

2.3 ALCUNI ESEMPI DI WMN TESTBED

Il testbed è un ambiente di esecuzione realizzato per verificare il funzionamento e le prestazioni di un prototipo hardware e/o software. Nell' ambito delle reti wireless l' uso dei testbed è spesso preferito a quello delle simulazioni, poiché queste si basano

su una rappresentazione semplificata del canale, che può condizionare l' affidabilità dei risultati dei test. Di seguito vengono descritti alcuni esempi di testbed reali: Roofnet [10] sviluppata dal M.I.T. di Boston, UCSB-MeshNet [16] dell' università di Santa Barbara, UMIC-Mesh [17] dell' università di Aachen.

Roofnet

Roofnet è una WMN utilizzata come base per la progettazione di protocolli di routing, di algoritmi di selezione del rate di trasmissione e di meccanismi per la valutazione della qualità del link, allo scopo di migliorare il throughpt offerto dalla rete stessa.

Roofnet è composta da 37 nodi e copre un' area di quattro chilometri quadrati a Cambridge, Massachusetts. I nodi sono dislocati sul territorio secondo un approccio unplanned, ossia la loro disposizione non è studiata a priori, ma è semplicemente basata sulla disponibilità di volontari, ai quali viene fornito accesso ad Internet, ad utilizzare l' hardware fornitogli. Ciascun nodo consiste di un PC, una scheda di rete wireless 802.11b e di un' antenna omnidirezionale collocata sul tetto. L' accesso ad Internet è fornito attraverso la porta Ethernet del PC.

Ogni PC è equipaggiato con una distribuzione Linux RedHat su cui viene eseguito Click Modular Router, un modulo software, sviluppato mediante l' omonimo framework, che ridefinisce le operazioni dello stack di protocolli di rete, sostituendosi a quello fornito dall' O.S.. In questo modulo viene definito un protocollo di routing apposito, l' SRCR, che è una versione migliorata dell' SrcRR [11], un meccanismo di valutazione della qualità del canale, l' Estimated Transmission Time (ETT) che, stimando il tempo di trasmissione tra ciascuna coppia di nodi, consente allo SRCR di determinare le rotte con throughput più alto (tempo di trasmissione totale stimato minimo).

L' analisi delle prestazioni ha messo in luce che, nonstante la natura unplanned della rete, questa riesca a fornire un discreto throughput e come i nodi siano ben serviti nonostante i pochi gateway presenti al suo interno.

UCSB MeshNet

La University of California Santa Barbara (UCSB) mesh testbed è una rete wireless-mesh sperimentale dislocata nel campus dell' universita di Santa Barbara. L' obiettivo del testbed è quello di valutare i protocolli progettati per la realizzazione delle funzionalità delle WMN. È stato progettato anche un sistema di supporto per la gestione, monitoraggio e visualizzazione dello stato dei nodi della rete; è stata posta particolare attenzione nella progettazione di questo servizio al fine di evitare che questo avesse un impatto sugli esperimenti svolti.

La rete consiste di 30 nodi, disposti sui 5 piani di un edificio del campus universitario: metà di questi sono costituiti da due Linksys WRT54G connessi tra loro ed equipaggiati con OpenWrt, una distribuzione Linux dedicata ai sistemi embedded; uno dei due appartiene efettivamente alla rete mesh, l' altro, connesso ad una rete wireless già esistente, serve per la gestione del nodo mesh stesso [1]. I nodi rimanenti sono degli small factor PC, dotati di una scheda di rete wireless 802.11b ed una Ethernet, su cui viene eseguita una distribuzione Linux. La scheda di rete wireless fornisce l' accesso alla rete mesh, mentre l' altra è utilizzata per gestire il nodo.

Tutti i nodi eseguono l' algoritmo di routing AODV come modulo di livello Kernel e sono configurati con un' interfaccia tramite la quale si connettono ad una infrastruttura di rete preesistente, che permette loro di scambiarsi informazioni di controllo senza che queste si aggiungano al traffico dati.

UMIC-Mesh

L' obiettivo dichiarato dai progettisti di UMIC-Mesh è quello di fornire un nuovo approccio alla realizzazione di un testbed per WMN. Si parte dalla duplice considerazione che se da un lato i testbed forniscono risultati con il più alto grado di realismo rispetto a tutti gli altri modelli di test (analisi teorica, simulazione, emulazione e virtualizzazione), dall' altro hanno diversi limiti: un controllo limitato sull' ambiente di test, la difficoltà di

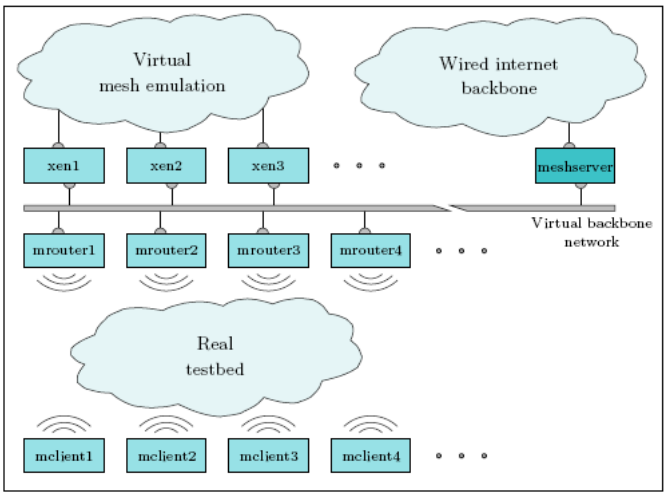

Figura 2: Architettura del testbed UMIC-Mesh

ripetere gli esperimenti e l' impossibilità di definire degli scenari di test complessi. Per risolvere questi problemi UMIC-Mesh si propone come un testbed ibrido, composto da una serie di nodi reali affiancati da un ambiente virutale. In figura 2 viene mostrato uno schema dell' architettura di rete.

L' ambiente di rete virtualizzato viene utilizzato principalmente per scopi di sviluppo del software e di validazione delle funzionalità: la configurazione centralizzata, basata sull' utilizzo di un MeshServer, semplifica la fase di distribuzione del software sui nodi della rete e la creazione degli scenari.

Il testbed è costituito da 48 single board PC dotate ciascuna di due schede di rete wireless 802.11a/b/g con chip Atheros AR5213 XR e due antenne omnidirezionali (mesh router): una antenna è utilizzata per comunicazioni router-to-router, l' altra per quelle router-to-client. Questi due tipi di comunicazione avvengono su canali differenti e secondo la modalità ahdemo, che omette la trasmissione dei messaggi di beacon e del BSSID. La rete virtuale è costituita da 7 PC, ognuno dei quali esegue fino a 10 virtual machine connesse tra loro da una virtual mesh emulation network. I mesh router e le virtual machine sono connesse tra loro attraverso una virtual backbone. Tale connessione è utilizzata per configurare i mesh router.

Il vantaggio principale legato all' utilizzo dell' approccio ibrido è dato dal fatto che il software sviluppato può essere prima testato sulla virtual mesh emulation network e successivamente essere installato sul testbed reale senza bisogno di apportare modifiche.

2.4 CENNI AGLI STANDARD APERTI

Nell' ambito delle comunicazioni wireless la definizione di standard aperti favorisce lo sviluppo di economie di scala e assicura l' interoperabilità tra le diverse soluzioni commerciali. Per rispondere a queste esigenze l' Institute of Electrical and Electronic Engineers (IEEE), organismo internazionale preposto alla standardizzazione delle specifiche nelle comunicazioni radio, predispone e coordina dei gruppi di lavoro, gli IEEE Task Group (TG), ognuno dei quali si occupa della definizione dei requisiti di una particolare tecnologia di comunicazione wireless emergente. Relativamente alle WMN, sono stati organizzati quattro TG che si sono occupati dell' estensione dei servizi di rete mesh nelle Wireless Personal Area Network (WPAN), nelle WLAN e nelle Wireless Metropolitan Area Network (WMAN). Da tale attività sono emersi quattro standard: l' 802.11*s* [4] per le WLAN, lo 802.15.5 [3] per le WPAN, l' 802.16a [5] per le WMAN e l' 802.20 [6] per l' accesso wireless a banda larga da parte di dispositivi mobili (MBWA)[2].

IEEE 802.15.5

Lo standard 802.15.5 definisce le specifiche per il livello fisico e MAC necessarie a supportare il dispiegamento di reti wireless mesh che coinvolgono un numero limitato di dispositivi mobili a corto raggio di copertura, come palmari e cellulari. Il motivo che ha condotto alla definizione di funzionalità mesh per le WPAN è legato alla limitata area di copertura offerta da questi dispositivi. L' adozione di un modello di comunicazione multihop aumenta sia la copertura fornita dal servizio di rete che la possibilità di utilizzare raggi di copertura ridotti che garantiscono throughput

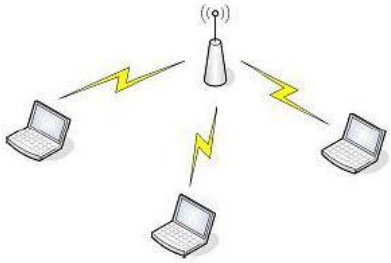

Figura 3: Infrastructured-based Wireless Network

più elevati.

IEEE 802.11s

L' 802.11 è lo standard per le WLAN. Oltre a definire le specifiche di base per l' architettura, l' 802.11 raccoglie, sottoforma di emendamenti, le diverse specifiche per le WLAN, dal supporto alla QoS (802.11e) a quello all sicurezza (802.11i) ecc.... La versione dello standard precedente all' emendamento sulle WMN definisce due architetture di rete possibili: le WLAN e le reti ad-hoc.

Le WLAN si basano su una topologia a stella in cui il nodo centrale, l' AP, fornisce l' area di copertura (detta anche Basic Service Set - BSS). L' AP annuncia la sua presenza attraverso la trasmissione periodica di messaggi di beacon. Ogni stazione che intende aggiungersi alla rete, deve effettuare l' accesso all' AP e se la rete è protetta deve autenticarsi nei suoi confronti. Una volta che la stazione si è connessa, tutti i pacchetti destinati ai nodi della WLAN o di altre reti raggiungibili devono transitare per l' AP vedi la fig. 3. Le WLAN possono essere collegate ad altre LAN, wireless o meno, attraverso la definizione di un Distribution System (DS), una rete fissa alle quale gli AP vengono collegati per realizzare un servizio di rete estesa. L' adozione di un' infrastruttura di rete cablata per l' estensione delle WLAN le rende poco adatte ad un loro impiego su larga scala.

Nella modalità ad-hoc, detta anche Independent Basic Service Set (IBSS), le stazioni possono comunicare direttamente tra di loro, senza bisogno dell' AP, vedi la fig. 4. Infatti, quando un nodo

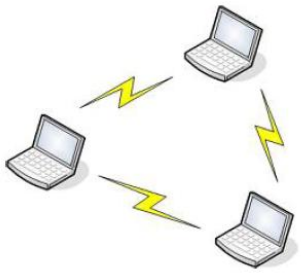

Figura 4: Ad-hoc Wireless Network

intende comunicare con altri nodi si pone in attesa di messaggi di beacon. Questi messaggi vengono inviati da quelle stazioni che annunciano la propria rete ad-hoc. Se il nodo riceve questo messaggio, si associa al nodo che lo ha trasmesso, realizzando una connettività punto-punto. Se non riceve alcun messaggio di beacon entro un certo intervallo di tempo, realizza lui stesso una rete ad-hoc, iniziando a trasmettere messaggi di beacon. Nonostante l' architettura decentralizzata le faccia assomigliare alle WMN, queste sono affette da due limiti che ne precludono un utilizzo in tal senso. Lo standard non definisce alcun servizio di estensione per questo tipo di reti. Inoltre ogni nodo può comunicare solo con quei nodi che ricadono nel suo radio range.

L' idea alla base dell' 802.11s, che ha raggiunto la fase finale di standardizzazione solo di recente, è quella di definire le specifiche di livello MAC che consentono di realizzare un Distribution System (DS) completamente wireless che, sfruttando il modello di comunicazione multihop e protocolli di routing appositi, sia dispiegabile su larga scala.

IEEE 802.16a

L' 802.16 fornisce l' insieme delle specifiche per il dispiegamento e funzionamento di una WMAN. Tra le altre cose questo standard definisce i protocolli per l' apertura delle connessioni primo/ultimo miglio alla WMAN. Una rete WMAN si basa su una architettura di comunicazione punto-multipunto (PMP), in cui una Base Station (BS) serve un certo numero di Subscriber Sta-

tion (SS) in una particolare area. La BS trasmette in broadcast a tutte le SS, mentre queste ultime utilizzano un link punto-punto per trasmettere alla BS.

L' utilizzo di alte frequenze ($>$ 10 GHz) per effettuare la trasmissione comporta che il sistema sia poco tollerante alle interferenze multipath. Per questo motivo le antenne di BS e SS devono trovarsi a vista (Line of Site - LOS). Per consentire una comunicazione che non necessiti del dispiegamento a vista delle antenne e che funzioni su frequenze libere è stato introdotto lo emendamento 802.16. L' adozione di funzionalità che permettono la comunicazione non-LOS (NLOS) ha permesso all' 802.16a di recepire le estensioni per definire le WMN.

IEEE 802.20

Il Gruppo opera per la standardizzazione dell' accesso mobile alla larga banda. Il MBWA 802.20 supporta il paradigma mesh per connessioni NLOS in scenari indoor/outdoor. A differenza dell' 802.16, l' 802.20 è stato definito per consentire a dispositivi che si muovono a velocità veicolare ($>$ 250 Km/h) di accedere alla rete. A tal proposito è stato definito un emendamento allo 802.16, l' 802.16e, allo scopo di fornire tale funzionalità anche nelle reti 802.16.

3

DESCRIZIONE DELLE TECNOLOGIE ADOTTATE NEL TESTBED DI SVILUPPO

L' ambiente utilizzato per i test sul modulo di link management consta di 5 router wireless dislocati nel Laboratorio Rosso, che si trova all' interno del dipartimento di Ingegneria Informatica, della Facoltà di Ingegneria dell' Università di Pisa. Questo testbed si basa su Roofnet [10], la WMN realizzata al MIT e utilizzata per fornire l' accesso ad Internet agli studenti che vivono in un' area della città di Cambridge vicino all' università.

Il capitolo si apre con la descrizione delle caratteristiche hardware (3.1) e software (3.2) dei dispositivi utilizzati nel nostro testbed. Nel sottoparagrafo 3.2.2 viene introdotto Click Modular Router, che oltre a fornire il framework di sviluppo utilizzato per implementare i servizi di rete sui dispositivi, rappresenta anche il modulo software che mette a disposizione le funzionalità sviluppate. Nel sottoparagrafo successivo (3.2.1) analizzeremo il meccanismo di assegnazione degli indirizzi alle interfacce di rete, grazie al quale possiamo aggiungere un nodo alla rete senza preoccuparci di configurarlo.

Nel paragrafo 3.5 verrà descritto il meccanismo di label switching, che ha sostituito il meccanismo di instradamento fornito da Roofnet; descriveremo poi (3.6) il protocollo di distribuzione delle label, grazie al quale è stato possibile automatizzare la installazione dei percorsi di rete basati sul label switching; e il meccanismo di fast rerouting (3.7), che fornisce un' intefaccia per l' attivazione di rotte di backup, da utilizzare non appena viene notificata l' interruzione della rotta principale; infine analizzeremo in dettaglio il protocollo di link management (3.8) secondo le specifiche definite nel draft 3.0 sull' 802.11s, che è stato utilizzato come base di riferimento per questo lavoro.

3.1 HARDWARE

Ogni nodo del testbed è costituito da un piccolo calcolatore su architettura x86, che disponde di:

- Due schede di rete wireless 802.11a/b/g, equipaggiate con antenne omnidirezionali.
- Due schede di rete Etherent per collegare i client alla WMN.

Le interfacce di rete wireless sono configurate con il meccanismo RTS/CTS disabilitato, essendone stata dimostrata l' inefficacia nell' evitare collisioni [10; 15], e con una modalità di funzionamento denominata "Pseudo-IBSS". Quest' ultima è simile allo standard 802.11 IBSS, in cui i nodi comunicano direttamente senza utilizzare gli AP, con la differenza che nella modalità pseudo-IBSS viene disabilitato lo scambio dei messaggi di beacon, di probe request/response e lo inoltro dell' informazione sul BSSID all' interno dei frame di livello MAC.[1] Questi due meccanismi sono stati disattivati perché quando la scheda di rete è impostata in modalità ad-hoc, i driver madwifi[2] causano la formazione di sotto-domini che, pur essendo relativi ad uno stesso SSID, hanno BSSID differenti.[3] Infatti benchè la ricerca di una rete ad-hoc venga fatta sulla base dell' SSID, il driver sottostante effettua il filtraggio dei pacchetti sulla base del BSSID. Il driver memorizza il BSSID relativo a quei pacchetti che per primi hanno annunciato il suo stesso SSID. Non essendoci alcuna corrispondenza tra SSID e BSSID, i pacchetti trasmessi da un altro nodo che annuncia lo stesso SSID verranno scartati perché associati ad un BSSID differente.

1 Il BSSID identifica il BSS/IBSS, ma è ottenuto dalla hash sull' indirizzo MAC associato all' interfaccia di rete dell' AP nel primo caso, del nodo che abilita la modalità ad-hoc nel secondo

2 Driver sviluppati per i chipset Atheros

3 L' SSID è una stringa impostata dall' utente che identifica, al pari del BSSID, il BSS/IBSS.

3.2 SOFTWARE E AUTOCONFIGURAZIONE

Ogni nodo esegue:

- Il sistema operativo OpenWrt [24], una distribuzione di Linux dedicata ai sistemi embedded.
- Il protocollo di routing, implementato in Click [19].
- Un server DHCP.

Dal punto di vista dell' utente il nodo offre una connessione immediatamente utilizzabile. È sufficiente infatti collegare il proprio PC alla porta Ethernet del nodo, che quest' ultimo assegna automaticamente un' indirizzo alla scheda di rete del PC mediante il server DHCP e configura se stesso come default IP router.

3.2.1 Indirizzamento

Come in Roofnet, un nodo trasmette i pacchetti IP incapsulandoli all' interno di un formato di messaggio appositamente definito per funzionare insieme al software di Roofnet; quindi, ogni nodo, sia a livello IP che a livello Roofnet, ha bisogno di essere identificato da un indirizzo univoco.

Il software in esecuzione su ciascun nodo deve essere in grado di assegnare un indirizzo al nodo stesso automaticamente, senza che vi sia bisogno di effettuare una configurazione esplicita. Ciò viene ottenuto costruendo un indirizzo IP, in cui i 24 bit meno significativi corrispondono ai 24 bit meno significativi dell' indirizzo MAC della prima interfaccia Ethernet e gli 8 bit più significativi corrispondono all' identificatore di rete di un indirizzo di classe A. Più in dettaglio, se X.X.X sono i 24 bit recuperati dall' indirizzo MAC, l' indirizzo identificativo del nodo sarà 5.X.X.X, mentre quelli delle interfacce wireless saranno 3.X.X.X (ath0) e 4.X.X.X (ath1); è importante notare che questi indirizzi hanno senso solamente per il protocollo di routing e per il livello Roofnet, quindi non sono utilizzabili in Internet [30].

Gli indirizzi delle interfacce Ethernet invece sono 1.0.0.1 (eth0) e 2.0.0.1 (eth1) e il DHCP assegna al client che si collega ad una

Interfaccia	Network	Indirizzo
eth0	1.0.0.0/8	1.0.0.1
eth1	2.0.0.0/8	2.0.0.1
ath0	3.0.0.0/8	3.X.X.X
ath1	4.0.0.0/8	4.X.X.X
srcr	5.0.0.0/8	5.X.X.X

Tabella 1: Assegnamento degli indirizzi alle interfacce di un nodo

di esse, rispettivamente, l' indirizzo 1.X.X.X o 2.X.X.X; in tal modo l' indirizzo del client identifica anche il nodo a cui è connesso.

3.2.2 Click Modular Router

Le funzionalità di rete installate sui router del testbed, ad eccezione dell' LDP, che è stato implementato come processo esterno a Click ed eseguito in spazio utente, sono state sviluppate utilizzando l' ambiente software Click Modular Router [19; 20; 21].

Click è composto da un framework per lo sviluppo di unità elementari per il processamento dei pacchetti (denominate elementi) e un modulo software che consente di istanziare e comporre tali funzionalità e renderle disponibili al router. Il funzionamento di Click prevede che il programmatore scriva un file di configurazione, un file di testo con estensione .click, all' interno del quale:

1. Dichiara quali elementi dovranno essere istanziati ed utilizzati da Click per manipolare i pacchetti in fase di esecuzione.

2. Specifica le connessioni tra gli elementi dichiarati, che definiscono le rotte di processamento dei pacchetti.

Completata la definizione dello script di configurazione, Click può essere eseguito con questa configurazione e il router messo in esecuzione, a meno di errori nella dichiarazione degli elementi o nella connessione degli stessi. In figura 5 viene mostrata una

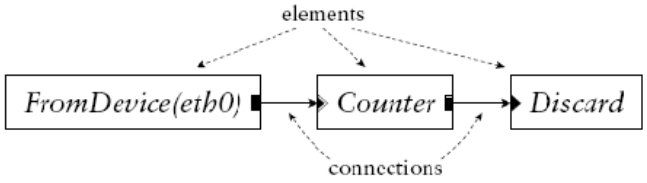

Figura 5: Una semplica configurazione in Click

rappresentazione grafica di una semplice configurazione. Gli elementi sono rappresentati come scatole e le connessioni tra gli elementi come frecce con verso.

Il software di Click è compilato sui router del testbed per essere eseguito come modulo del kernel (vedi fig. 6). In questo modo Click si sostituisce al tradizionale meccanismo di manipolazione dei pacchetti fornito dal kernel, con il risultato che i pacchetti vengono processati rapidamente, in quanto il loro recupero avviene direttamente dal dispositivo e non è mediato dallo stack dei protocolli di rete. L' altra modalità di funzionamento prevede che Click venga compilato ed eseguito come processo utente (vedi fig. 7); in tal caso Click può processare i pacchetti solo dopo averli catturati dallo stack di rete. A causa di questo doppio passaggio le code del kernel smaltiscono i pacchetti molto più lentamente, sperimentando così frequenti fenomeni di saturazione.

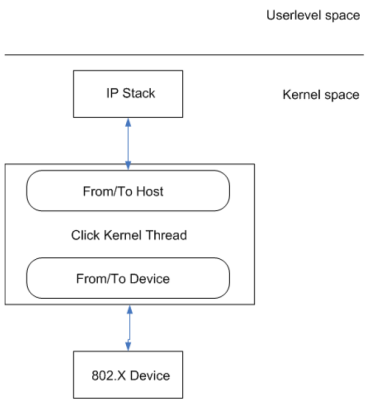

Figura 6: Schema generale di Click in modalità kernel level

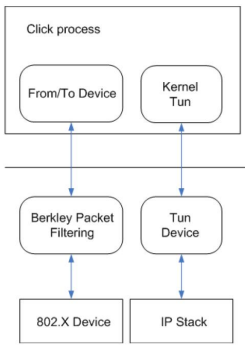

Figura 7: Schema generale di Click in modalità user level

Click mette a disposizione un insieme di elementi già definiti per la creazione di apposite configurazioni.[4] Le proprietà più importanti di un elemento sono:

- Element class: ogni elemento appartiene ad una element class; questa specifica il codice da eseguire quando l'elemento stesso processa un pacchetto.
- Port: ogni elemento ha un certo numero di porte di ingresso e di uscita. Ogni connessione unisce una porta di uscita di un elemento con una porta di ingresso di un altro elemento. Ne esistono tre tipi: push, pull e agnostic.
- Configuration string: contiene degli argomenti addizionali, separati da virgole, che viene passata ad un elemento al momento dell' inizializzazione.
- Method interface: ogni elemento supporta una o più method interface tramite le quali comunicare tra loro a tempo di esecuzione.
- Handler: ad ogni elemento si possono associare un certo numero di handler, ovvero punti di accesso per l'interazione con l'utente piuttosto che con altri elementi. Click in modalità kernel utilizza il filesystem `/proc` per comunicare con i processi-utente. Nel caso in cui Click sia eseguito

4 la lista completa degli elmenti è disponibile all' indirizzo http://www.read.cs.ucla.edu/click/elements

come processo-utente, gli handler possono essere acceduti attraverso l' interfaccia socket.

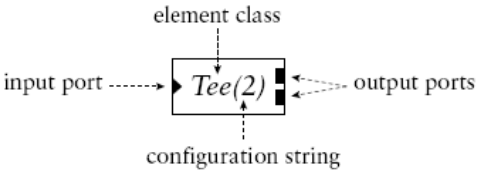

Figura 8: Un semplice elemento

In figura 8 vengono mostrate le proprietà sopra elencate e associate ad un semplice elemento. *Tee* è il nome della classe C++ che deriva dalla classe base *Element* e definisce l' elemento; un elemento *Tee* copia ciascun pacchetto ricevuto dall' unica porta d' ingresso su ciascuna delle porte d' uscita (in questo caso sono due). Le porte di ingresso sono rappresentate con dei triangoli, quelle di uscita con dei rettangoli. Le porte di colore nero, indicano delle porte di tipo push, mentre quelle di colore bianco delle porte di tipo pull. I metodi di interfaccia e gli handler non sono specificati esplicitamente poiché sono definiti all' interno della classe dell' elemento.

In Click è anche possibile definire nuovi elementi, qualora quelli esistenti non forniscano le funzionalità richieste. Un nuovo elemento viene implementato come una classe C++ che estende la classe base `Element`; eredita da questa alcuni metodi/oggetti e ne ridefinisce altri per fornire le funzionalità specifiche del nuovo elemento. Per maggiori dettagli sulle modalità con cui si definiscono nuovi elementi o su come si costruiscono i file di configurazione si rimanda all' ampia documentazione on-line [21; 22; 23].

In figura 19 viene mostrata la rappresentazione grafica della configurazione in Click dei router del testbed. *EthDevice* e *WirelessDev* sono i blocchi che trasmettono/ricevono i pacchetti alle/dalle rispettive interfacce fisiche.[5] Il blocco *LinuxHost* non corrisponde ad alcuna interfaccia fisica; bensì rappresenta una interfaccia virtuale per l' invio/ricezione di pacchetti verso/dal-

5 In Click i blocchi tratteggiati indicano elementi composti, costruiti all' interno dello script e ottenuti dalla composizione di elementi di base scritti in C++

lo stack di protocolli di rete del sistema operativo. Tranne il blocco LinuxHost, tutti gli altri blocchi associati alle interfacce fisiche forniscono i pacchetti ai blocchi a valle incapsulandoli in frame di tipo ethernet.

I pacchetti, indipendentemente dall' interfaccia di arrivo, vengono inviati ai blocchi a valle incapsulati in frame di tipo ethernet e, dopo essere stati contrassegnati con l' identificativo della interfaccia di arrivo dai blocchi *Paint*, trasmessi al blocco *Mesh router*.[6] Quest' ultimo è il blocco che implementa le funzionalità del routing e quelle del forwarding. In ingresso al blocco i frame vengono discriminati sulla base del campo *ethernet type* e inviati sul ramo di processamento corrispondente, distinguendo, ad esempio, il ramo del routing da quello del calcolo delle statistiche sulla bonta dei link, a quello che si occupa dell' instradamento del traffico dati. Il traffico, dati e di controllo, che deve essere gestito dai WirelessDev, viene trasmesso dal Mesh router su due connessioni separate. Questo avviene perché i blocchi WirelessDev schedulano la trasmissione delle due tipologie di traffico dando una priorità più alta a quello di controllo.

3.3 PROTOCOLLO DI ROUTING: SRCR

L' SRCR [10], anch' esso sviluppato per Roofnet, è un protocollo di routing che ricerca i percorsi con più alto throughput tra ogni coppia di nodi. L' utilizzo di antenne omni-direzionali consente all' SRCR di avere la visione di molti più link e quindi di scegliere tra questi quelli a throughput maggiore.

Il protocollo si ispira al Dynamic Source Routing (DSR) per il processo di routing. Il source-routing è una tecnica di routing in cui il nodo sorgente determina in anticipo la sequenza completa di nodi da attraversare e la specifica nell' intestazione dei messaggi che trasmette; ciò al fine di evitare i problemi di routing-loop che si presentano con i meccanismi di forwarding classici e causati dai cambiamenti della metrica associata ai link.

6 l' identificativo dell' interfaccia d' arrivo corrisponde al primo byte dello indirizzo IP associato all' interfaccia medesima

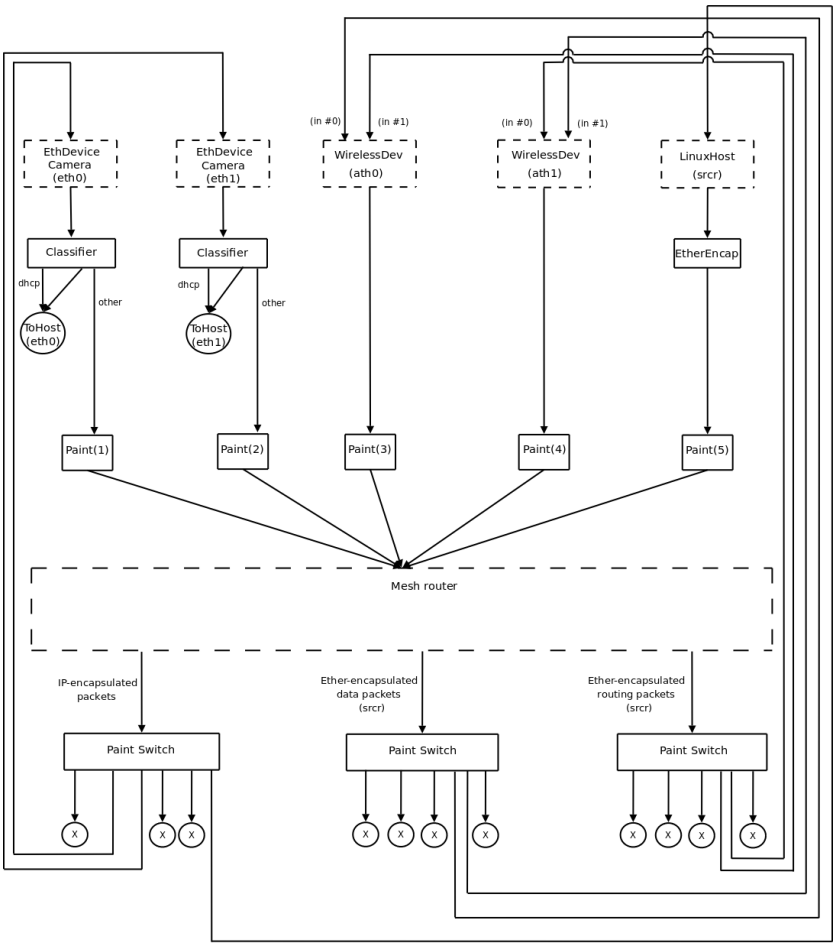

Figura 9: Schema a blocchi della configurazione Click associata ad ogni router

In SRCR ciascun nodo mantiene un database parziale con le metriche dei link presenti tra ogni coppia di nodi; il nodo esegue l' algoritmo di Dijkstra su questo database, ricavando le rotte migliori verso tutte le destinazioni note. I nodi SRCR vengono a conoscenza dei link e delle metriche ad esse associate in tre modi diversi:

- Il nodo che instrada un pacchetto su di un link, include la metrica corrente associata a quel link all' interno del campo source-route; in questo modo tutti i nodi lungo la rotta possono acquisire informazioni sulla metrica.

- Se un nodo deve instradare un pacchetto per il quale non conosce la source-route trasmette in flooding un messaggio di route request (come avviene con il Route Discovery Protocol del DSR).

- Gli altri nodi intercettano i messaggi di route-request e di route-response e memorizzano le mtriche specificate nei pacchetti di risposta all' interno del proprio database.

Inoltre i nodi impostati come gateway utilizzano un meccanismo che contribuisce all' aggiornamento dei database delle rotte sui vari nodi. Ciascun gateway trasmette periodicamente e in flooding una dummy-query che raggiungendo tutti i nodi, trasporta con se l' informazione su tutti i link attraversati e le metriche ad essi associate. Viceversa, quando un nodo un pacchetto dati ad un gateway è quest' ultimo che acquisisce informazioni di routing. Questo consente di ridurre il più possibile il numero di volte in cui un nodo invia in flooding una route request.

L' inconveniente legato allo schema finora visto è che non è detto che le route-request seguano sempre il percorso migliore per raggiungere il destinatario della richiesta. Per risolvere questo problema l' approccio seguito è il seguente: quando un nodo riceve una nuova query, prima memorizza all' interno del proprio database la sequenza di link e la metrica specificate nel campo source route, dopodiché esegue Dijkstra, calcolando il percorso migliore a partire dal nodo che ha trasmesso la query, infine memorizza il percorso appena calcolato nel campo source-route (sostituendo il precedente valore), infine il nodo invia la query in broadcast. Se in un secondo momento il nodo riceverà un' altra copia della query, che gli consente di ottenere una rotta con una metrica migliore rispetto a quella calcolata in precedenza, allora instraderà nuovamente la query specificando la nuova rotta. Dagli studi su Roofnet si è visto che in una rete più grande e densa, nella quale i nodi mandano dati a molte destinazioni, il meccanismo sopra descritto non è scalabile.

Il cambiamento dello stato e/o della metrica dei link può comportare un aggiornamento delle rotte utilizzate. Se un link associato ad una rotta attiva non è più accessibile, il nodo upstream lo notifica a tutti i mittenti la cui trasmissione è fallita. Se la

metrica associata ad una rotta attiva peggiora, il mittente che utilizza quel link apprenderà la nuova metrica del link dai pacchetti che lo attraversano. Un nodo sorgente esegue l' algoritmo di Dijkstra tutte le volte che viene a conoscenza del cambiamento della metrica associata ad un link; in questo modo egli utilizza sempre le rotte migliori che ha a disposizione in un dato momento.

3.4 METRICHE

Il protocollo di routing SRCR sceglie le rotte migliori sulla base della metrica *Estimated Transmission Time* (ETT), derivata dalla metrica *Estimated Transmission Count* (ETX) [13]. La metrica ETX, proposta da De Couto, stima il numero di ritrasmissioni necessario per inviare correttamente i pacchetti unicast, andando a misurare il tasso di perdita (loss rate) di pacchetti broadcast scambiati tra coppie di nodi vicini. Il limite principale della metrica ETX è quello di non tenere conto del diverso rate di trasmissione che c' è tra i link. Infatti dall' analisi delle prestazioni di Roofnet [10], si è notato che spesso link con un tasso di perdita elevato (anche del 50%), forniscono in ogni caso un throughput maggiore di link con tassi di perdita inferiore, ma con rate di trasmissione più bassi. Va notato infatti che da un lato i rate di trasmissione variano con una granularità grossa (nelle schede 802.11b, 1Mbps, 2Mbps, 5.5Mbps, 11Mbps), dall' altro il processo di ritrasmissione viene eseguito in pochi millisecondi.

ETT stima il tempo totale necessario a trasmettere un pacchetto su una rotta composta da n link come:

$$ETT = \sum_{i=1}^{n} ETX_i * \frac{s}{b_i} \tag{3.1}$$

dove ETX_i corrisponde alla probabilità di consegna corretta di un pacchetto tra due nodi , s è la dimensione del pacchetto, b_i corrisponde bit-rate nominale associata al link e s/b_i è uguale al tempo di trasmissione che ha prodotto quel dato valore di ETX_i. SRCR sceglie le rotte con ETT minore, in altre parole quelle che offrono un tempo di trasmissione minore.

SRCR utilizza la seguente procedura per calcolare la probabilità di consegna corretta ETX_i associata a ciascun link: ciascun nodo trasmette periodicamente e in broadcast messaggi di 1500 byte a tutti i rate b_i possibili e, sempre periodicamente e in broadcast, pacchetti di 60 byte con un tasso di 1Mbps; con la trasmissione dei primi viene simulata la trasmissione dei pacchetti dati, con i secondi quella degli ack.[7] I nodi tengono traccia della frazione dei messaggi broadcast che ricevono da ogni nodo vicino e riportano tali statistiche al vicino corrispondente. La probabilità di consegna corretta dal nodo A a quello B, ad un determinato bit-rate, è pari al prodotto della frazione di pacchetti da 1500 byte inviati correttamente da A a B (e di cui B ha informato A) e della frazione di pacchetti di 60 byte ricevuti da B.

SRCR calcola la metrica ETT_i associata al link i come il minimo prodotto tra la probabilità di consegna corretta di un pacchetto (ETX_i) e il bit-rate (s/b_i) che ha prodotto tale valore di probabilità. Riferendoci all' equazione 3.1 questa può essere riscritta in termini di throughput nel modo seguente:

$$t = \frac{1}{\sum_i \frac{1}{t_i}} \tag{3.2}$$

che esprime la relazione che c' è tra il throughput totale t di una rotta e i throughput associati ai link che la compongono. Questa relazione è sufficientemente accurata nel caso di rotte composte da pochi link, mentre è poco accurata nel caso di rotte che coinvolgono molti nodi.

3.5 LABEL SWITCHING

Il meccanismo d' instradamento dei dati originariamente presente sui router del testbed e basato sul source routing dei pacchetti è stato sostituito con un meccanismo di label switching conforme all' MPLS (Multiprotocol Label Switching) [18] e defi-

7 I pacchetti broadcast non vengono ritrasmessi in caso di perdita da parte del livello MAC dell' 802.11. Ciò significa che la frazione di pacchetti broadcast ricevuti da un nodo in un determinato intervallo di tempo consente di stimare il numero di ritrasmissioni di pacchetti unicast da parte del meccanismo ARQ dell' 802.11

nito in un precedente lavoro di tesi [1]. Questo meccanismo è stato definito come estensione software di Click.

L' MPLS definisce un modello di instradamento dei dati che opera ad un livello intermedio tra i livelli 2 (DataLink layer) e 3 (Network layer) del paradigma OSI. La natura Multiprotocol dell' MPLS si riferisce al fatto che il suo funzionamento è indipendente rispetto al tipo di protocollo di livello dataLink e di livello network utilizzato.

Il processo di forwarding tradizionale si basa su un sistema distribuito per l' instradamento dei pacchetti; ogni router, in questo modello, decide autonomamente quale deve essere il next-hop a cui consegnare il pacchetto ricevuto. Nelle reti IP questa decisione viene presa sulla base del solo indirizzo IP destinatario, che rappresenta la chiave per l' accesso alla tabella di forwarding. In particolare ogni dispositivo analizza l' header di livello network del pacchetto, individua il next-hop basandosi sulle informazioni recuperate dall' algoritmo di routing e infine inoltra il pacchetto. L' obiettivo dell' MPLS è quello di evitare di analizzare il network header del pacchetto su ogni router incontrato lungo il percorso, attraverso la realizzazione di un nuovo meccanismo per la scelta del next-hop.

Il processo di selezione del next-hop può essere considerato come la composizione di due funzioni: la prima partiziona lo insieme di tutti i possibili pacchetti in un insieme di FEC (Forwarding Equivalence Class), mentre, la seconda funzione effettua la associazione tra ognuna di queste FEC ed un next-hop. In questo modo possiamo avere pacchetti totalmente differenti mappati sulla stessa FEC non risultando più distinguibili per un router. Con riferimento al routing IP tradizionale, ogni router lungo il percorso del pacchetto dovrà eseguire entrambe le funzioni, considerando due pacchetti come appartenenti alla stessa FEC solo se la parte network dell' indirizzo di destinazione di entrambi individua una stessa entrata della tabella di forwarding.

In MPLS invece, l' assegnamento di un pacchetto ad una determinata FEC è fatto solo dal router di ingresso al dominio MPLS e solamente una volta. Una FEC viene definita utilizzando un insieme di informazioni contenute in ciascun pacchetto che nel caso più semplice corrispondono al solo indirizzo destinatario; ad

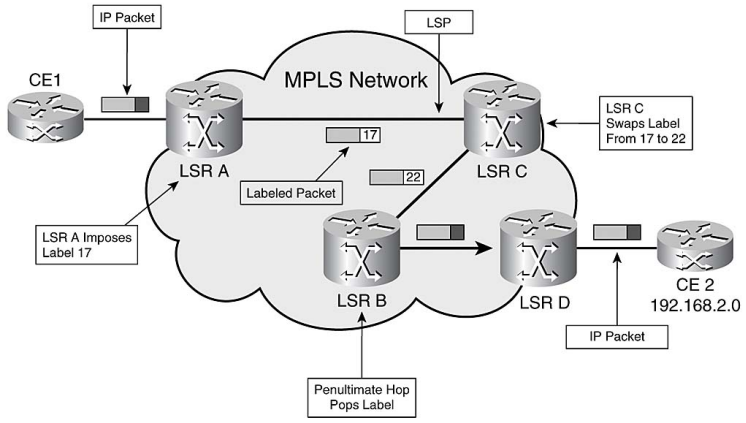

Figura 10: Struttura di una rete MPLS

esempio possiamo definire una FEC come l' insieme dei pacchetti aventi la stessa coppia indirizzo sorgente-destinazione, oppure come l' insieme di pacchetti associati ad una stesso indirizzo e porta sorgente ecc...

Per evitare che tutti gli LSR[8] valutino la FEC di appartenenza di un pacchetto per ricavarne il next-hop, l' ingress router[9] codifica la FEC stessa con un' informazione più corta e di dimensione fissa, la *label*, che viene aggiunta in testa al pacchetto. I restanti LSR non dovranno analizzare l' intestazione dei pacchetti per determinare la FEC di appartenenza, ma utilizzeranno la label come indice per l' accesso ad una tabella che contiene il next-hop per quella FEC e un nuovo valore della label, che sostituirà quello presente nel pacchetto ricevuto prima che questo sia inviato al next-hop.

I percorsi di rete che i pacchetti seguono in un dominio MPLS prendono il nome di Label Switched Path (LSP); questi partono da un ingress LSR e arrivano fino ad un egress LSR[10]. In figura 10 viene mostrato un esempio di dominio MPLS.

8 Label Switching Router: router che supporta le funzionalità di label switching dell' MPLS

9 Ingress LSR: router di bordo che si trova all' ingresso del dominio MPLS

10 Egress LSR: router di bordo di uscita da un dominio MPLS, che si occupa di rimuovere la label e instradare i pacchetti secondo il meccanismo di forwarding tradizionale

Ogni LSR gestisce tre tabelle, necessarie a fornire il meccanismo di label switching:

FTN (FEC-TO-NHLFE). Questa tabella associa ogni FEC ad una entrata della NHLFE che verrà utilizzata per l' instradamento dei pacchetti. La FTN viene usata sull' ingress LSR per determinare la FEC di appartenenza dei pacchetti privi di label che arrivano sul router.

ILM (INCOMING LABEL MAPPING). La ILM serve a gestire i pacchetti in ingresso ad un LSR che trasportano una label; questa tabella associa ad una label in ingresso una entrata della tabella NHLFE utilizzata per il forwarding.

NHLFE (NEXT-HOP LABEL FORWARDING ENTRY). Quest' ultima tabella è utilizzata quando si tratta di dover effettuare il forwarding di un pacchetto per il quale è stata trovata una corrispondenza nella FTN o nella ILM; ogni entrata di questa tabella contiene il next-hop a cui trasmettere il pacchetto, la label di uscita e infine l' azione da effettuare sul pacchetto, che può essere di *push*, ossia viene aggiunta una label in testa al pacchetto se esso proviene dalla FTN, o in testa allo stack di label, di *replace*, che rimpiazza la label più esterna associata al pacchetto con la label di uscita, di *pop*, che rimuove la label più esterna dal pacchetto.[11]

Nella figura 11 viene schematizzato il processo di forwarding operato dai LSR.

3.6 LABEL DISTRIBUTION PROTOCOL

Per sfruttare le potenzialità di un sistema di label switching occorre un protocollo di configurazione e scambio delle label che permetta ad un LSR di aprire un LSP in maniera automatica. La soluzione adottata in questo lavoro [30] ed installata sui router del testbed si basa sulla scelta dell' LDP come protocollo di distribuzione delle label.

11 il label stack è un meccanismo che consente di aggiungere una label ad un pacchetto che ne possiede già una o più di una. Questa tecnica viene sfruttata per implementare il meccanismo di fast rerouting

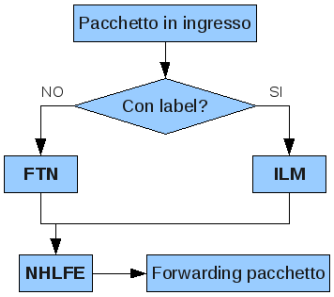

Figura 11: Schema relativo al processo di intradamento dell' MPLS

Due LSR che usano LDP per scambiare informazioni sulle associazioni label/FEC sono identificati come LDP peer; le informazioni vengono scambiate attraverso una sessione LDP stabilita tra i due. Le funzioni principali svolte dall' LDP sono quelle di:

- Neighbor discovery.
- Apertura e mantenimento delle sessioni.
- Label advertisement.
- Notifica degli errori.

Ogni LSR esegue un processo di discovery che gli consente di localizzare LSR vicini, direttamente connessi a livello link. La procedura di discovery è basata sull' invio periodico di messaggi di *LDP Link Hello*, trasportati in pacchetti UDP e destinati, a livello IP, al gruppo multicast a cui i router sono registrati. La ricezione di un messaggio di Hello sull' interfaccia di un LSR consente a questo di individuare una Hello Adjency nei confronti del LSR che ha inviato il messaggio.

Stabilita una HA, i due nodi aprono una sessione; l' apertura di una sessione avviene in due fasi: prima viene aperta una connessione TCP, dopodiché vengono negoziati i parametri della sessione (fase di inizializzazione). Una volta stabilita la sessione, questa viene mantenuta attiva attraverso lo scambio periodico di messaggi di *Keepalive* all' interno della connessione TCP. Lo utilizzo di una connessione TCP per lo scambio dei messaggi è dovuto alla necessità di garantirne la sequenza logica tra trasmissione e ricezione. A questo punto i due LSR sono da considerare

LDP peer e possono iniziare a creare e distribuire associazioni tra FEC/label (fase di label advertisement).

Relativamente alla gestione delle label, LDP distingue tra modalità di distribuzione e politche di memorizzazione/mantenimento delle label. Le corrispondenze FEC/label possono essere annunciate secondo una modalità *downstream-on-demand* (DoD), in base alla quale le label sono associate ad una particolare FEC dall' LSR a valle (downstream LSR), ma solo su richiesta dello LSR a monte (upstream LSR); quest' ultimo infatti trasmette un messaggio di *label request* che, seguendo il percorso calcolato dal protocollo di routing sottostante, raggiungerà l' LSR next-hop per quella FEC. Questo LSR risponderà con un messaggio di *Label Mapping* che, ripercorrendo la rotta a ritroso, installerà l' LSP sugli LSR attraversati. Alternativamente non appena un LSR stabilisce una nuova associazione FEC/label può annunciarla a tutti gli LSR adiacenti inviando dei messaggi di label mapping; questa modalità prende il nome di *unsolicited downstream* (UD).

Entrambe le due modalità di distribuzione sopra descritte possono operare alternativamente in *independent control mapping* oppure in *ordered control mapping*. Nel primo caso ogni LSR assegna e distribuisce le corrispondenza FEC/label in modo indipendente dagli altri, in risposta o ad una nuova FEC aggiunta nella routing table (UD), o ad una richiesta di label mapping fatta da un LSR upstream (DoD), per la quale il nodo ha quella FEC nella propria routing table. Nel secondo caso la distribuzione del label mapping avviene in maniera ordinata dall' egress LSR verso l' ingress LSR. Ogni LSR prima di annunciare il proprio label mapping al LSR a monte deve attendere il label mapping dal LSR a valle.

La memorizzazione delle label può avvenire secondo un approccio *liberal label retention mode*, che consente ad un LSR di mantenere nella sua FTN tutte le corrispondenze FEC/label, anche se sono state annunciate da LSR che non sono suoi next-hop per le FEC considerate. Nell' approccio *conservative retention mode* un LSR memorizza esclusivamente le coppie FEC/label ricevute da LSR che sono suoi next-hop per quelle FEC.

Il software LDPL (Label Distribution Protocol Light) [30] è stato realizzato come processo utente eseguito da ciascun LSR; è

stata preferita questa implementazione a quella ad elementi integrata in Click, perché a livello kernel non è possibile gestire timer e connessioni TCP. LDPL implementa un sottoinsieme delle funzionalità relative al sistema di distribuzione e conservazione delle label; fornisce infatti la funzionalità di distribuzione DoD con ordered control mapping. Il servizio di memorizzazione delle label è stato realizzato secondo la modalità conservative.

LDPL utilizza le rotte fornite dall' SRCR per instradare i messaggi di label request. Inoltre monitora periodicamente la tabella di routing al fine di rilevare eventuali cambiamenti di rotta per una certa destinazione; in tal modo l' LDPL può decidere se installare un nuovo LSP che segua la nuova rotta.

3.7 FAST REROUTING

Il Fast Rerouting è una tecnica che, in presenza di guasti o interruzioni ad un percorso di rete principale, provvede a ridurre i tempi di attesa dovuti al disservizio. Essa si basa sull' installazione di LSP di backup alternativi agli LSP principali. L' LSR che ha a disposizione entrambi gli LSP, mantiene i label mapping relativi ai due percorsi all' interno della propria tabella NHLFE; l' FRR database, un database delle rotte di backup, conserva la informazione che associa un LSP al suo LSP di backup. Sarà il modulo di link management che, attraverso dei meccanismi per la rilevazione della qualità dei link o dei nodi vicini, attiverà lo LSP di backup qualora quello principale non sia più utilizzabile.

L' LSP di backup può essere installato staticamente dall' amministratore di rete, il quale dovrà specificare l' intero percorso di rete che tale LSP attraverserà, oppure dinamicamente attraverso delle procedure che, in fase di installazione dell' LSP principale, individuano ed installano i percorsi di backup.

La procedura di individuazione ed installazione degli LSP di backup, denominati *Detour*, è stata realizzata come estensione del software LDPL. L' LDPL fornisce un meccanismo originale per l' installazione dei detour. Lo scopo è quello di individuare ed utilizzare delle deviazioni dall' LSP principale in modo da saltare il punto di interruzione e ritornare sullo stesso LSP a par-

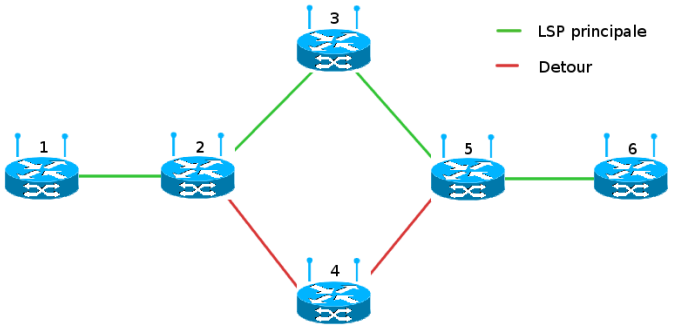

Figura 12: Un esempio di detour: si noti come il detour protegga sia da un eventuale interruzione del link tra LSR2 e LSR3 che dal guasto di LSR3

tire dal nodo successivo al guasto. A causa di questa particolare formulazione, un detour è sempre di lunghezza pari a due nodi. L' installazione dei detour viene realizzata in fase di apertura degli LSP principali, per sfruttare la disponibilità di informazioni di routing necessarie alla loro definizione. In fig. 12 viene mostrato un esempio di detour installato in un dominio MPLS. Per maggiori dettagli sulle caratteristiche del protocollo LDP si rimanda al lavoro [30].

3.8 LINK MANAGEMENT

Realizzare una configurazione mesh in un contesto wireless richiede che ogni nodo aggiunto alla WMN gestisca esplicitamente i collegamenti con i nodi vicini. Per ottenere ciò, nell' 802.11s viene definita una macchina a stati finiti (MSF) e delle procedure specifiche che consentono a ciascun nodo di aprire dei link verso i vicini; una volta aperti, i link vengono monitorati dai nodi ad essi associati per ricavarne informazioni sulla qualità; Tali informazioni vengono scambiate ed utilizzate da ciascun nodo per decidere se il link in questione può essere coinvolto nel processo di forwarding.

Con riferimento al draft 3.0 [29] dell' 802.11s indicheremo con *Mesh Basic Service Set* (MBSS) la WMN. I nodi che partecipano al MBSS vengono chiamati *mesh STA*. Mesh STA vicine, individuate a seguito della fase di discovery (3.8.1), prendono il nome di

Candidate Peer mesh STA. I link, risultato dell' esecuzione della macchina a stati tra due nodi (3.8.2), sono indicati con il termine *peering*. Due mesh STA che hanno aperto un peering tra loro sono chiamate *Peer mesh STA*.

I messaggi utilizzati per gestire il processo di discovery e di peering management sono:

- Frame di discovery: beacon e/o probe request/response.
- Frame di peering management: open, confirm, close.

I beacon sono frame trasmessi in broadcast a livello MAC e vengono utilizzati da una mesh STA per annunciare la sua presenza/appartenenza all' MBSS e i parametri di configurazione adottati per l' accesso. I frame di probe request sono trasmessi in broadcast e vengono inviati da una mesh STA che intende accedere ad un particolare MBSS; l' obiettivo è quello di cercare mesh STA già associate a quel MBSS. Se una mesh STA associata all' MBSS riceve un messaggio di probe request per quel MBSS, risponderà con un frame di probe response trasmesso in unicast. Questo frame trasporta le stesse informazioni contenute nel frame di beacon.

I frame di peering management vengono inviati da una (candidate) peer mesh STA per effettuare l' apertura, mantenimento e la chiusura dei peering verso altre (candidate) peer mesh STA. Tutti i frame di peering management sono trasmessi in unicast.

3.8.1 Mesh discovery

Il processo di peering management (3.8.2) richiede che una mesh STA ottenga informazioni sulle mesh STA vicine. Il processo di discovery si basa sull' utilizzo di beacon o di probe request. Quando una mesh STA viene accesa essa può utilizzare una procedura di scanning attivo, inviando frame di probe request, o passivo, ponendosi in attesa di messaggi di beacon, per individuare la presenza di un particolare MBSS.

Mesh ID

L' 802.11s introduce un nuovo identificatore, il *Mesh ID*, allo scopo di identificare l' MBSS. Il Mesh ID è costituito da una stringa di caratteri e viene inserito all' interno dei frame di discovery. Nel caso dei messaggi di beacon/probe response per annunciare l' MBSS di appartenenza; nel caso dei frame di probe request per ricercare un particolare MBSS al quale la mesh STA intende associarsi. La funzione del Mesh ID è simile a quella svolta dall' SSID[12], utilizzato per identificare le WLAN, con la differenza che il Mesh ID previene la possibilità che una STA (un nodo non-mesh) possa associarsi ad una mesh STA che non fornisce il servizio di AP (non-AP mesh STA). Per lo stesso motivo una non-AP mesh STA non annuncerà nei beacon/probe response un SSID valido, utilizzabile dalla STA per associarsi ad essa.

Profili mesh

Una mesh STA deve supportare il concetto di profilo mesh. Un profilo mesh consiste di:

1. un Mesh ID
2. un *path selection protocol identifier*: identificatore del protocollo di routing adottato all' interno dell' MBSS
3. un *path selection metric identifier*: identificatore del tipo di metrica adottato all' interno dell' MBSS
4. un *congestion control mode identifier*: identificatore del meccanismo di controllo della congestione adottato nello MBSS
5. un *synchronization protocol identifier*: identificatore del protocollo di sincronizzazione adottato nell' MBSS
6. un *authentication protocol identifier*: identificatore del protocollo di autenticazione adottato nell' MBSS

Il profilo mesh associato ad un MBSS viene annunciato nei frame di beacon e di probe response trasmessi dalle mesh STA presenti nell' MBSS. Ad una mesh STA è possibile associare più profili mesh, ma soltanto uno può essere attivo.

12 Service Set ID

Candidate peer mesh STA discovery

Lo scopo di questa procedura è quello di scoprire le candidate peer mesh STA associate ad un MBSS e le relative configurazioni. Questa procedura viene eseguita da una mesh STA sia prima che dopo la sua associazione ad un MBSS. Se la mesh STA è già membro di un MBSS, ha esattamente un profilo attivo. Quando una mesh STA disattiva il proprio profilo attivo, le informazioni di sessione ad esso legate vengono disabilitate.

Una mesh STA esegue una procedura di scanning attivo o passivo per individuare la presenza di mesh STA vicine. Una mesh STA scoperta deve essere considerata una candidate peer mesh STA se e soltanto se:

1. Viene ricevuto un frame di beacon o di probe response proveniente dalla mesh STA scoperta.
2. Il frame di beacon o di probe response ricevuto contiene un Mesh ID uguale a quello della mesh STA che effettua lo scanning (scanning mesh STA).
3. Il frame di beacon o di probe response ricevuto contiene un *information element* (vedi fig. 18) di tipo Mesh Configuration (vedi par. 3.8.3) che specifica:
    a. Un numero di versione di configurazione supportato.
    b. Un path selection protocol id. ed un path selection metric id. uguali a quelli associati alla scanning mesh STA.
    c. Un congestion control mode id. uguale a quello associato alla scanning mesh STA.
    d. Un synchronization protocol id. uguale a quello associato alla scanning mesh STA.
    e. Un authentication protocol id. uguale a quello associato alla scanning mesh STA.
    f. Il campo Accepting Peerings, contenuto a sua volta all' interno del campo Mesh Capability, impostato ad 1.

In altre parole la mesh STA scoperta può essere considerata candidate peer se il profilo che annuncia corrisponde ad uno dei

profili associati alla scanning mesh STA, se questa non ha ancora un profilo attivo, o all' unico profilo attivo, se essa fa già parte di un MBSS.

3.8.2 Mesh peering management

Descrizione Generale

Il protocollo di mesh peering management è utilizzato per stabilire, mantenere, e chiudere peering tra mesh STA. Due mesh STA non devono potersi scambiare frame che non siano quelli di discovery e di peering management finché non hanno stabilito un peering tra loro.

Dopo aver scoperto una candidate peer mesh STA, la mesh STA può iniziare il protocollo di peering management per aprire un peering verso di lei. La SME (Station Management Entity), l' entità che controlla la mesh STA, utilizza il Peering Instance Controller (PIC) per eseguire il protocollo stesso.

Il PIC gestisce i tentativi di apertura di un peering associandogli una struttura dati denominata *peering instance*. Ogni peering instance è identificata univocamente dalla quadrupla <*localMAC*, *peerMAC*, *localLinkID*, *peerLinkID*>. Il localMAC rappresenta l' indirizzo MAC della mesh STA. Il peerMAC rappresenta l' indirizzo MAC della candidate peer mesh STA o della peer mesh STA. Il LocalLinkID e il peerLinkID sono due numeri interi generati rispettivamente dalla mesh STA e dalla candidate peer o peer mesh STA. Il localLinkID deve essere unico tra tutti gli identificatori locali associati ai peering aperti sulla mesh STA. Lo stesso vale per il peerLinkID dal lato della candidate peer o peer mesh STA. Il peerLinkID deve essere specificato dalla candidate peer o peer mesh STA all' interno del frame di peering open e peering confirm che trasmette. Il PIC consente a due mesh STA di stabilire al più una peering instance, anche se in fase di apertura permette di avere più peering instance aperte; quando una di esse completa l' apertura, tutte le altre dovranno essere chiuse.

La mesh STA può iniziare il protocollo di peering management in due modi: passivo o attivo. Nel primo caso, la mesh STA si pone in attesa di frame di peering open provenienti da

un candidate peer mesh STA. Nel secondo caso, la mesh STA crea essa stessa una peering instance e contestualmente trasmette un messaggio di peering open destinato alla candidate peer mesh STA verso la quale si intende aprire il peering.

Una peering instance termina quando il peering viene chiuso. Tra le cause che determinano la chiusura di un peering ci sono la corretta ricezione di un frame di peering close oppure l' errato processamento di un frame di peering management.

Il protocollo di peering management utilizza tre timer per controllare il comportamento del protocollo stesso. Ad ogni peering instance vengono associati tre timer:

- retryTimer: definisce il tempo massimo che una mesh STA è disposta ad attendere per la ricezione del frame di peering confirm, in risposta ad uno dei frame di open da lei inviato in questa istanza del protocollo. Scaduto questo timer la mesh STA ha a disposizione un certo numero di tentativi per l' apertura di quel peering, superato il quale il tentativo di apertura del peering viene chiuso.

- confirmTimer: limita il tempo massimo che una mesh STA attende per la ricezione di un frame di peering open dopo aver già ricevuto un frame di peering confirm dalla candidate peer mesh STA.

- holdingTimer: limita il tempo massimo che una mesh STA attende per la ricezione di un frame di peering close dopo aver già trasmesso un frame di chiusura per quella peering instance.

Nel diagramma di flusso in figura 13 viene illustrata l' interazione tra il processo di discovery e quello di peering management. Per completezza viene mostrata anche l' interazione con il protocollo di autenticazione, anche se questo non verrà preso in considerazione nella fase di implementazione del modulo di link management. Se non è richiesta autenticazione per l' apertura del peering, la mesh STA inizia il protocollo di peering management (PLM). Se il protocollo di peering management fallisce, la mesh STA torna ad eseguire la fase di discovery; altrimenti viene creata una sessione tra le due peer mesh STA.

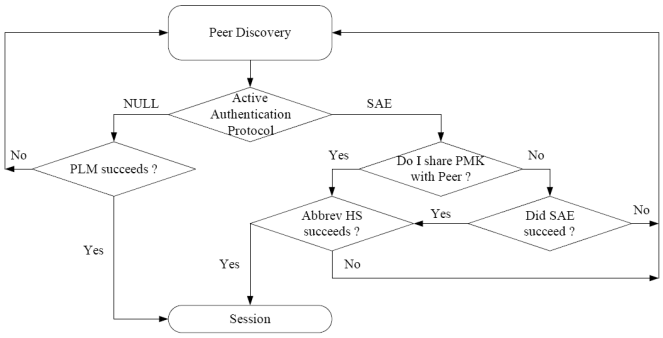

Figura 13: diagramma di flusso che descrive l' interazione tra il protocollo di discovery e quello di peering management

Processamento dei frame di peering management

Come già è stato detto all' inizio del paragrafo, la mesh STA utilizza i frame di peering management per aprire un peering verso una candidate peer mesh STA. Un frame di open serve a richiedere l' apertura di una peering instance da parte della mesh STA che lo trasmette nei confronti di quella che lo riceve. Ogni frame di open specifica un insieme di parametri di configurazione per la peering instance; in particolare il frame deve contenere:

- Un information element di tipo *Mesh Peering Protocol Version*.
- Il Mesh ID.
- Un information element di tipo Mesh Configuration. Questo deve specificare, in più rispetto alle informazioni che trasporta all' interno del frame di beacon/probe response, il valore del campo MCCA Enabled, in base alla configurazione specificata per la mesh STA.
- Un information element di tipo Peering Management information element, il quale deve specificare il campo LocalLinkID scelto dalla mesh STA che ha trasmesso il frame.
- Altri information element, in relazione all' insieme di protocolli configurati sulla mesh STA.

La mesh STA che riceve il frame di open lo processa verificando che:

- il Mesh Peering Protocol Version sia impostato al valore <ANA 42>, che attesta che il messaggio è relativo al protocollo di peering management.
- il Mesh ID annunciato sia uguale a quello specificato dal proprio profilo attivo.
- il Mesh Configuration specifichi dei parametri compatibili rispetto a quelli del proprio profilo attivo. L' insieme minimo di parametri che deve essere verificato è lo stesso visto nel caso della procedura di candidate peer mesh STA discovery; è possibile aggiungerne altri, per i quali si rimanda alla documentazione relativa [29].

La mesh STA deve scartare il frame di open se una sola delle verifiche sopra descritte dovesse fallire e chiude il tentativo di apertura del peering inviando un frame di peering close. Altrimenti, il frame viene accettato; in tal caso viene aggiornato lo stato della peering instance e viene memorizzata la parte di identificatore della peering instance relativa alla mesh STA che ha trasmesso il frame di apertura.

La mesh STA che ha accettato il frame di open invia anch' essa un frame di apertura, nel quale specifica per la sua parte le informazioni sopra elencate. Dopodiché la mesh STA trasmette un frame di peering confirm per dare riscontro alla candidate peer mesh STA della corretta ricezione del frame di open da lei inviato. Nel frame di conferma viene specificato l' identificatore completo della peering instance e vengono ripetute le informazioni di configurazione (Mesh Peering Protocol Version, Mesh ID e Mesh Configuration information element) associate al proprio profilo attivo. Tali informazioni verranno nuovamente verificate dalla mesh STA ricevente. Il doppio processamento di queste informazioni viene fatto allo scopo di assicurare che le due mesh STA siano d' accordo sul profilo scelto.

La fase di apertura di un peering si conclude con successo se vengono soddisfatte le seguenti condizioni:

1. Entrambe le mesh STA hanno inviato e ricevuto (e correttamente processato) un frame di peering open relativo al peering considerato.

2. Entrambe le mesh STA hanno inviato e ricevuto (e correttamente processato) un frame di peering confirm relativo allo stesso peering.

La procedura di chiusura di un peering prevede che la mesh STA invii un frame di close alla peer mesh STA o alla candidate peer mesh STA per chiudere il peering o un tentativo di apertura di peering. Il frame di chiusura deve contenere:

- il Mesh Peering Protocol Version: deve essere impostato al valore <ANA 42>.

- il Mesh ID: deve contenere il Mesh ID associato al proprio profilo attivo.

- il Mesh Peering Management: contiene l' identificatore completo se alla peering instance è già associato l' identificatore completo. In caso contrario deve specificare il solo LocalLinkID.

La mesh STA che riceve il frame di chiusura lo scarterà se il valore del Mesh ID è diverso da quello specificato dal proprio profilo attivo. Altrimenti, la mesh STA accetta il frame di peering close e dopo aver inviato anch' essa il frame di close chiuderà la peering instance in questione.

Ad ogni peering instance viene associata una MSF (fig. 14) che ne gestisce apertura, mantenimento e chiusura. Ecco una breve descrizione degli stati:

- IDLE: stato iniziale/finale. In questo stato la MSF è pronta per aprire, passivamente o attivamente, una nuova peering instance.

- OPN_SNT: la MSF ha inviato un frame di peering open, secondo la procedura di apertura attiva, e si pone in attesa di un frame di peering open e di peering confirm da parte della candidate peer mesh STA.

- CNF_RCVD: la MSF ha ricevuto un frame di peering confirm, ma non quello di peering open. Quindi la MSF non ha ancora trasmesso il frame di peering confirm.
- OPN_RCVD: in questo stato la MSF ha ricevuto il frame di apertura, ma non quello di conferma. La MSF ha anche inviato il frame di conferma in risposta al frame di apertura ricevuto.
- ESTAB: In questo stato la MSF ha trasmesso e ricevuto entrambi i frame di apertura e chiusura. Il peering è ora utilizzabile per lo scambio del traffico dati tra le due peer mesh STA.
- HOLDING: In questo stato la MSF sta chiudendo la peering instance stabilita con una peer mesh STA o il tentativo di peering in atto con una candidate peer mesh STA.

3.8.3 Struttura dei frame

Esistono quattro tipi di frame distinti sulla base dei valori del campo *type* interno al campo di *frame control* (vedi fig. 16):

- Management
- Control
- Data
- Reserved

I tipi di frame sopra elencati sono caratterizzati da un numero minimo di campi in comune (vedi fig. 15). I primi tre campi (*Frame Control*, *Duration/ID*, e *Address* 1) e il campo finale *FCS* costituiscono la struttura minima di ogni frame. La presenza degli altri campi dipende dal tipo e sottotipo di frame.

I frame utilizzati dalla procedura di discovery e dal protocollo di peering management sono tutti di tipo management (fig. 17); il campo *subtype* (vedi fig. 16) consente di distinguere tra loro i frame appartenenti allo stesso tipo. Nella tabella 2 vengono mostrati i valori di *type* e *subtype* specificati dallo standard 802.11 [9]

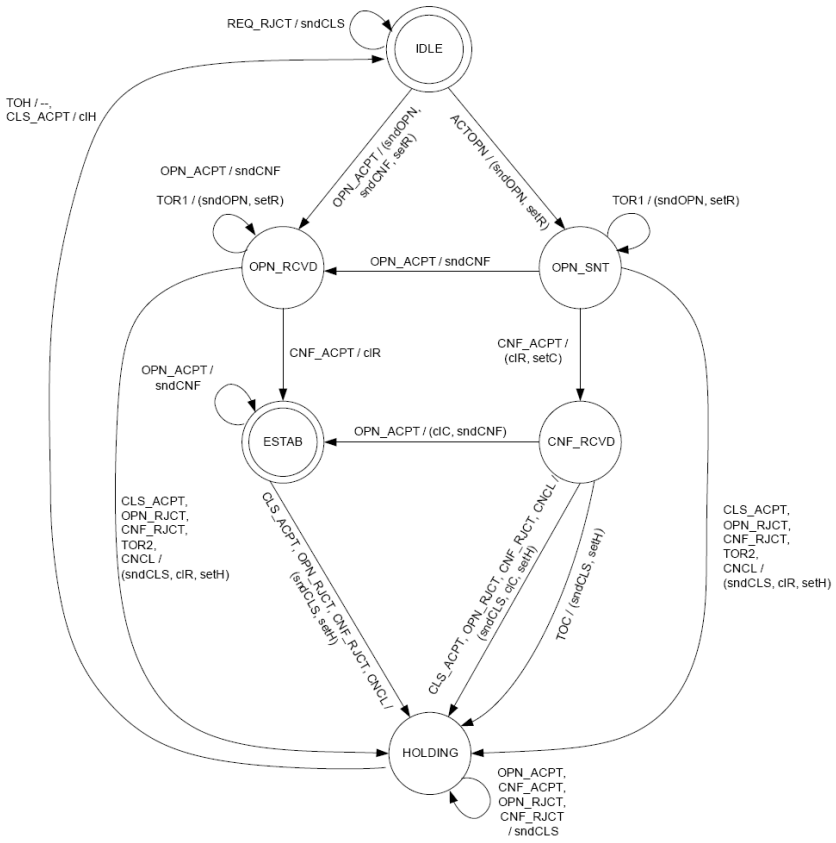

Figura 14: Macchina a stati finiti relativa al protocollo di peering management

per distinguere tra loro probe request, beacon, probe response e frame di peering management (*Action subtype*).

I frame appartenenti al sottotipo action, nel nostro caso i frame di open, confirm e close, sono distinti tra loro per mezzo dello omonimo campo che segue quello di *Frame Control*. È possibile estendere il formato dei frame grazie agli information element (vedi la fig. 18). Questi hanno una struttura generale caratterizzata da 1 byte per il campo *Element ID*, che identifica il tipo di

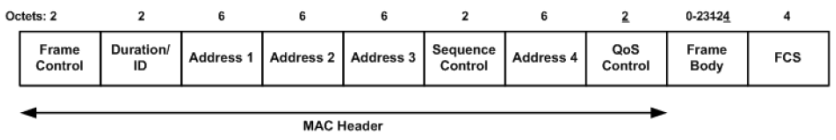

Figura 15: Struttura generale di un frame 802.11

Type value b3 b2	Type description b7 b6 b5 b4	Subtype value	Subtype description
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	1000	Beacon
00	Management	1101	Action

Tabella 2: Combinazioni di valori type e subtype

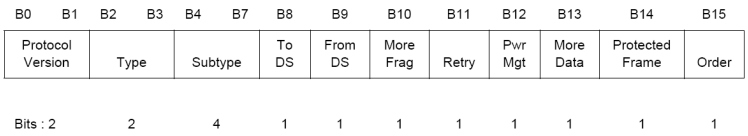

Figura 16: Struttura del campo frame control

elemento, 1 byte per il campo *Length*, che indica la dimensione del campo che segue, il campo *Information*.

Con riferimento ai protocolli di discovery e di peering management, gli information element incontrati sono: il Mesh ID, il Mesh Peering Protocol Version, il Mesh Peering Management e il Mesh Configuration. Per maggiori informazioni riguardo al contenuto dei frame di beacon, probe request/response e di peering management si rimanda al draft 3.0 [29].

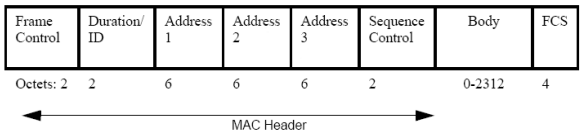

Figura 17: Struttura del frame di management

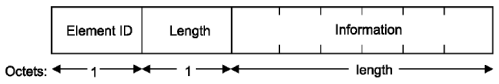

Figura 18: Struttura di un information element

4

IMPLEMENTAZIONE DEL LINK MANAGEMENT CON CLICK MODULAR ROUTER

Il modulo di link management è stato implementato attraverso la definizione di un unico elemento in Click, l' Sme; questo fa uso di alcune strutture dati per gestire configurazioni, profili e istanze di peering associate alla mesh STA.

Il capitolo si apre (4.1) con l' esposizione delle modifiche fatte allo script di configurazione per supportare l' Sme. Nel paragrafo successivo (4.2) vengono analizzate le strutture dati definite nel modulo di link management e da questo utilizzate per gestire le funzionalità offerte. Si passa poi (4.3) ad illustrare in dettaglio l' implementazione delle funzionalità fornite dall' Sme. Nel paragrafo (4.4) viene descritta la lista degli argomenti che compongono la stringa di configurazione, con cui viene inizializzato l' Sme; segue poi (4.5) la sinossi dei comandi forniti dagli handler, accompagnata da una breve descrizione degli stessi. Nello ultimo paragrafo vengono descritte le soluzioni adottate per integrare il funzionamento dell' Sme con l' quello dell' infrastruttura di routing preesistente.

4.1 MODIFICHE ALLO SCRIPT

All' interno del modulo di link management l' Sme è l' unica classe C++ che deriva dalla classe base `Element`. L' elemento Sme ha due porte di ingresso e 4 di uscita di tipo push; Come si vede in figura 19, le porte di uscita dei blocchi WirelessDev sono connesse alle porte di ingresso del blocco Sme che hanno lo stesso indice. Delle 4 porte di uscita, le prime due, quelle di indice 0 e 1, sono collegate con i blocchi Paint(3) e Paint(4), le ultime due sono connesse alle due porte di ingresso dei WirelessDev che gestiscono il traffico di controllo. Questo significa che nessun altro elemento interno allo script sarà coinvolto nel processamento e

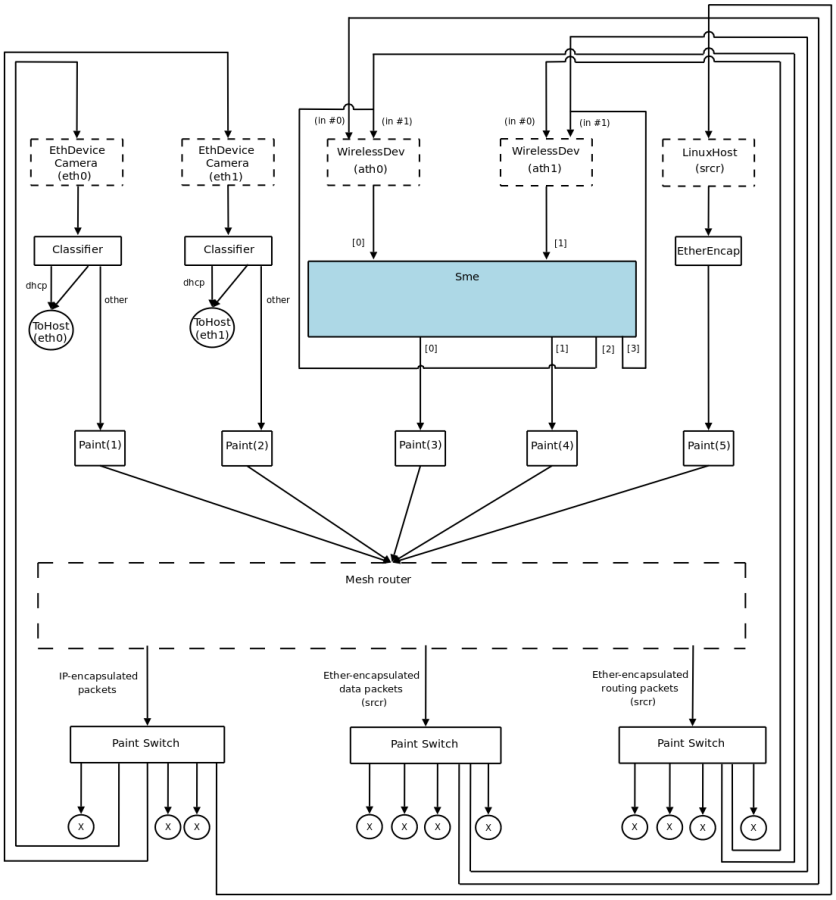

Figura 19: Schema a blocchi finale della configurazione Click

instradamento dei frame di controllo prodotti dall' Sme. Si noti che per ragioni di compatibilità con l' architettura preesistente lo Sme trasmette i messaggi incapsulandoli come corpo dei frame di tipo Ethernet.

L' Sme intercetta tutto il traffico ricevuto dalle due interfacce wireless; di questo, quello destinato all' elemento viene processato al suo interno, mentre quello destinato al Mesh router viene inoltrato sulle porte 0 e 1 soltanto se l' indirizzo MAC sorgente si riferisce ad una mesh STA vicina verso la quale è già stato stabilito un peering; ciò al fine di rispettare il vincolo definito in [29], secondo il quale due mesh STA che non abbiano stabilito un peering tra loro non possono scambiarsi alcun tipo di traffico se non quello prodotto dalle rispettive Sme.

4.2 STRUTTURE DATI

Ciascuna mesh STA è caratterizzata da:

- Una o più configurazioni associate alla STA, indipendentemente dal fatto che partecipi o meno ad una WMN.
- Una o più configurazioni associate alla mesh STA, delle quali solo una può essere attiva.
- Uno o più profili, dei quali uno solo può essere attivo
- Uno o più candidate peer, ciascuno dei quali ha associato il profilo da lui annunciato e accettato dalla mesh STA.
- Una o più peering instance, ognuna delle quali rappresenta un peering stabilito o in fase di apertura tra due peer o candidate peer mesh STA.
- Uno o più indirizzi mac associati alle interfacce di rete di mesh STA vicine, verso le quali si vuole inibire l' apertura dei peering.
- Un Sme che esegue il processo di discovery e la procedura di installazione, mantenimento e rimozione di peering instance verso le peer o candidate peer mesh STA.

Le informazioni associate ai primi sei punti sono definite come entrate di vettori gestiti secondo una politica FIFO. I vettori definiti sono: la *STAConfig*, la *MeshSTAConfig*, la *MeshProfile*, la *CandidatePeer*, la *PeeringInstance* e la *Blacklist*; queste strutture dati sono state implementate utilizzando una versione di STL [22] definita da Click e resa disponibile anche al livello kernel.

L' Sme è la struttura dati che svolge il ruolo di gestore della mesh STA; in particolare, fornisce le funzionalità richieste dal servizio mesh, mantiene i vettori sopra elencati e permette ad un utente di accedervi attraverso gli handler.

4.2.1 STAConfig

La classe STAConfig implementa un vettore di *STAConfigEntry*, che consente di associare una configurazione ad ognuna del-

le schede di rete wireless presenti sul router. I campi contenuti nella classe STAConfigEntry sono stati scelti sulla base delle specifiche fornite dall' allegato D della bozza 3.0 [29].

I campi che definiscono una STAConfigEntry sono:

- *StationID*: corrisponde all' indirizzo MAC dell' interfaccia a cui è associata la configurazione; rappresenta la chiave per discriminare due entry tra di loro.
- *BeaconPeriod*: questo campo specifica l' intervallo di tempo base utilizzato dall' Sme per calcolare il periodo effettivo associato alla trasmissione dei messaggi di beacon (lo intervallo effettivo è scelto come multiplo di quello base).
- *MeshEnabled*: indica se sull' interfaccia di rete è abilitato il servizio mesh.

I metodi principali implementati dalla classe STAConfig sono: la *add_entry*, che aggiunge una STAConfigEntry, passata come argomento; la *remove_entry*, che rimuove un' entrata dal vettore, specificando come argomento il campo StationID oppure l' entrata stessa; *lookup_entry*, che effettua una ricerca lineare di una entry, specificando come argomento la StationID che identifica la entrata da cercare; la *print_table*, che visualizza il contenuto dell' intero vettore.

Attualmente la configurazione delle interfacce wireless può avvenire solo all' interno del metodo *configure* definito per l' elemento Sme; i parametri per ciascuna entry vanno specificati allo interno della stringa di configurazione fornita con la dichiarazione dell' Sme. È necessario configurare almeno un' interfaccia e non si possono configurare più di due interfacce.

4.2.2 MeshSTAConfig

Questa struttura dati definisce un vettore di *MeshSTAConfigEntry* e consente di specificare una o più configurazioni relative ad uno stesso Mesh ID e quindi ad una stessa WMN, ma anche a mesh ID differenti. Solo una configurazione per volta può essere attiva; tale configurazione è significativa sull' intera mesh STA,

ossia tutte le sue interfacce wireless abilitate al servizio mesh opereranno con la medesima configurazione.

Il contenuto della entry è conforme alle specifiche fornite dallo allegato D ed è caratterizzato dai seguenti campi:

- *MeshID*: questo attributo definisce il valore del Mesh ID associato a questa entrata.
- *MeshForwarding*: specifica se la mesh STA è abilitata a trasmettere i frame.
- *MeshMaxRetries*: specifica il numero massimo di frame di open che possono essere. trasmessi per effettuare l' apertura di un peering.
- *MeshRetryTimeout*: specifica il valore del timeout, in millisecondi, associato al Retry Timer.
- *MeshTTL*: specifica il valore del campo TTL impostato dalla mesh STA sorgente.
- *MeshConfTimeout*: specifica il valore del timeout, in millisecondi, associato al Confirm Timer.
- *MeshHoldingTimeout*: specifica il valore del timeout, in millisecondi, associato all' Holding Timer.
- *Active*: indica se questa MeshSTAConfigEntry è attiva oppure no.

Le entry sono discriminate sulla base dei valori associati a tutti i campi. Questo consente di specificare due entry che differiscono per il valore di un solo campo. Come la STAConfig , anche la MeshSTAConfig fornisce i metodi per aggiungere, rimuovere o cercare un' entrata all' interno del vettore e un metodo per stamparne tutto il contenuto. In questo caso però, non essendo presente un attributo-chiave nella entry, i primi tre metodi richiedono come argomento l' entrata da aggiungere, rimuovere o ricercare. Tali metodi sono accessibili anche via handler. È possibile specificare una entry, attiva o meno, nella stringa di configurazione; se ciò non accadesse l' elemento Sme verrebbe inizializzato con una MeshSTAConfigEntry di default disattive. L'

attivazione di una configurazione, se non effettuata dall' utente, avviene a seguito dell' accettazione di un profilo pubblicizzato dai frame della fase di discovery (vedi oltre 4.2.3).

4.2.3 MeshProfile

Questa struttura dati definisce un vettore di *MeshProfileEntry* attraverso le quali specifica uno o più profili associabili ad uno stesso o a diversi Mesh ID. Solo un profilo per volta può essere attivo e questo è significativo per tutte le interfacce wireless della mesh STA. I campi della classe MeshProfileEntry sono:

- *MeshID*: definisce il valore del Mesh ID associato a questa entrata.
- *PathSelectionProtocolID*: identificatore del protocollo di routing supportato da questo Mesh ID e disponibile a livello data link.
- *PathSelectionMetricID*: identificatore della metrica supportata da questo Mesh ID e disponibile a livello data link.
- *CongestionMode*: identificatore del meccanismo di controllo della congestione definito per questo Mesh ID e disponibile a livello data link.
- *SynchronizationProtocolID*: identificatore del protocollo di sincronizzazione associato a questo Mesh ID e disponibile a livello data link.
- *AuthenticationProtocolID*: identificatore del meccanismo di autenticazione associato a questo Mesh ID e disponibile a livello data link.
- *AcceptPeering*: specifica se la mesh STA supporta o meno la apertura di peering verso di lei
- *Active*: specifica se il profilo in questione è attivo o meno.

Le entrate sono discriminate sulla base dei valori associati a tutti gli attributi, consentendo così di specificare due entry che

differiscono per il valore di uno solo di essi. I metodi per aggiungere, rimuovere o cercare un' entrata richiedono come argomento l' entrata stessa. Questo perché, come per la tabella MeshSTAConfig, le entrate non hanno un attributo-chiave. È disponibile un metodo per visualizzare il contenuto di MeshProfile. Tali metodi sono accessibili anche via handler.

È possibile specificare una entry, attiva o meno, nella stringa di configurazione; se ciò non accadesse l' elemento Sme verrebbe inizializzato con una MeshProfileEntry di default disattiva. L' attivazione di un profilo, se non esplicitamente indicato dallo utente, viene effettuata durante la fase di discovery. Se l' interfaccia di una mesh STA con profilo disattivo riceve un frame di probe response o di beacon che pubblicizza lo stesso profilo, essa attiva il proprio se è dotata di una configurazione che lo supporti. In tal caso, anche la configurazione verrà attivata. Il profilo viene annunciato dalla mesh STA all' interno dei messaggi di beacon, di probe response, di open e di confirm.

4.2.4 CandidatePeer

Questa classe definisce un vettore di *CandidatePeerEntry*, attraverso il quale mantiene una lista delle candidate peer mesh STA individuate e il profilo da esse annunciato. In particolare la classe *CandidatePeerEntry* è caratterizzata dai seguenti campi:

- *CandidatePeerMAC*: indirizzo MAC associato all' interfaccia della candidate peer mesh STA di cui la mesh STA ha accettato il profilo annunciato.
- *LocalMAC*: indirizzo MAC dell' interfaccia locale associata al candidate peer.
- *MeshID*: Mesh ID associato al profilo annunciato e accettato.
- *PathSelectionProtocolID*: identificatore del protocollo di routing associato al profilo annunciato e accettato.
- *PathSelectionMetricID*: identificatore della metrica associata al profilo annunciato e accettato.

- *CongestionMode*: identificatore del meccanismo di controllo della congestione associata al profilo annunciato e accettato.
- *SynchronizationProtocolID*: identificatore del meccanismo di sincronizzazione annunciato e accettato.
- *AuthenticationProtocolID*: identificatore del protocollo di autenticazione annunciato e accettato.
- *Forwarding*: specifica se la candidate peer mesh STA effettua l' instradamento dei frame.
- *AcceptPeering*: specifica se la candidate peer mesh STA accetta l' apertura di peering in ingresso.
- *BeaconReceived*: contatore del numero di frame di beacon ricevuti.
- *LastBeaconReceived*: marca temporale che indica l' istante in cui è stato ricevuto l' ultimo frame di beacon.

La coppia di indirizzi MAC costituisce la chiave per le entrate del vettore.[1] Come le altre classi, anche CandidatePeer definisce dei metodi per aggiungere, rimuovere o cercare un' entrata allo interno del vettore e un metodo per stamparne tutto il contenuto. In particolare è possibile effettuare la ricerca/cancellazione di una entry sulla base della coppia di indirizzi MAC, passata come argomento, oppure sulla base dell' indirizzo MAC passato come primo argomento, che viene interpretato come indirizzo del candidate peer o locale in base alla stringa passata come secondo argomento. Lo stesso vale per i metodi di ricerca di una entrata.

4.2.5 PeeringInstance

La classe definisce un vettore di *PeeringInstanceEntry*, che vengono istanziate dall' Sme per gestire l' apertura, mantenimento

1 Questo è vero nel caso generale in cui un router ha più interfacce wireless configurate sullo stesso canale; diversamente l' indirizzo MAC del candidate peer è di per sé chiave

e chiusura di un peering. Una PeeringInstanceEntry è composta dai seguenti campi:

- *LocalLinkID*: numero pseudo-casuale scelto dalla mesh STA per identificare questo peering.
- *PeerLinkID*: numero pseudo-casuale scelto dalla peer o candidate peer mesh STA per identificare questo peering.
- *LocalMAC*: indirizzo MAC dell' interfaccia locale alla mesh STA al quale è stato associato questo peering.
- *PeerMAC*: indirizzo MAC dell' interfaccia locale alla peer o candidate peer mesh STA al quale è stato associato questo peering.
- *LocalAssociationID*: numero pseudo-casuale scelto dall' Sme locale per identificare la peer o candidate peer mesh STA.
- *PeerAssociationID*: numero pseudo-casuale scelto dall' Sme della peer o candidate peer mesh STA per identificare la mesh STA.
- *EventType*: indica qual è stato l' ultimo evento prodotto dall' esecuzione dell' istanza di MSF su questo peering.
- *PeeringState*: indica lo stato in cui si trova la particolare istanza di MSF in relazione all' apertura/mantenimento o chiusura di questo peering.
- *MetricFromMe*: specifica l' ultimo valore della metrica associato a questo peering, nel verso che va dalla mesh STA alla peer mesh STA.
- *MetricToMe*: specifica l' ultimo valore della metrica associato a questo peering, nel verso che va dalla peer mesh STA alla mesh STA.
- *AverageMetricFromMe*: specifica il valor medio della metrica, calcolato come media mobile su un intervallo di dieci campioni di MetricFromMe.

- *AverageMetricToMe*: specifica il valor medio della metrica, calcolato come media mobile su un intervallo di dieci campioni di MetricToMe.
- *Item*: individua qual è il tipo di timer impostato attualmente per questo peering.
- *Start*: specifica l' istante in corrispondenza del quale è stato aperto questo peering.
- *Timeout*: intervallo di tempo con cui viene caricato il timer attualmente impostato.
- *EstabInterval*: specifica l' intervallo di tempo impiegato per stabilire questo peering.
- *Retries*: corrisponde al numero di messsaggi di open ritrasmessi nel tentativo di aprire questo peering. Questo valore non può superare il valore *MaxRetries* definito dalla configurazione (MeshSTAConfigEntry) attiva sulla mesh STA.
- *Config*: puntatore alla MeshSTAConfigEntry attiva per questo peering.
- *Profile*: puntatore alla MeshProfileEntry attiva per questo peering.

Le PeeringInstanceEntry possono essere discriminate tra loro sulla base di diversi sottoinsiemi di campi. In particolare, le combinazioni di campi che costituiscono una chiave per le entrate di questa tabella sono:

- <localMAC, localLinkID, peerMAC, peerLinkID>
- <localMAC, localLinkID>
- <peerMAC, peerLinkID>
- <localMAC, peerMAC>

Il motivo per cui anche la coppia di indirizzi MAC è chiave sarà chiaro quando andremo ad analizzare le scelte implementative adottate per l' Sme.

La PeeringInstance fornisce dei metodi per creare (*add_entry*), rimuovere (*remove_entry*) o cercare (*lookup_entry*) una PeeringInstanceEntry. Sono state definite diverse versioni di *lookup_entry* legate alle possibili chiavi di ricerca a disposizione.

4.2.6 Blacklist

Questa classe realizza un vettore di indirizzi MAC relativi ad interfacce di mesh STA vicine, verso le quali l' Sme bloccherà la apertura dei peering. La classe fornisce un metodo per aggiungere o rimuovere un indirizzo MAC e per visualizzare tutto il contenuto del vettore.

4.2.7 Sme

La parte privata della classe Sme, oltre a specificare un istanza per ognuna delle strutture dati definite sopra, contiene i seguenti campi:

- *ScanMode*: indica la modalità di scanning configurata per la Sme. Valori possibili sono: *Active* e *Passive*.

- *OpenMode*: indica la modalità di apertura configurata per la Sme. Valori possibili sono: *Active* e *Passive*.

- *PollTimer*: polling timer che scatena l' esecuzione periodica del metodo *run_timer*, il cui compito è quello di schedulare la trasmissione di messaggi di probe request o di beacon; inoltre gestisce gli eventi periodici relativi all' esecuzione della MSF associata ad ogni peering instance; tali eventi corrispondono all' esaurimento del timer (retry, confirm, holding) impostato su ciascuna peering instance. Il Polling Timer è quello che viene rischedulato con frequenza maggiore.

- *ExpireTimeout*: intervallo di tempo con cui Click rischedula la esecuzione del metodo *run_timer*. È dimensionato sullo ordine dei millisecondi.

- *ShortTermTimer*: Timer di breve termine che, attraverso la funzione *handle_st_expire*, esegue periodicamente il metodo *remove_expired_entries*; quest' ultimo è utilizzato dall' Sme chiude le peering instance, rimuove i link di livello SRCR e le candidate peer/peer mesh STA dalle quali non riceve un messaggio di beacon da un certo intervallo di tempo.

- *ShortTermTimeout*: intervallo di tempo con cui Click rischedula l' esecuzione della *handle_st_expire*. È dimensionato sullo ordine dei secondi.

- *CpDeadInterval*: intervallo di tempo in secondi utilizzato dalla remove_expired_entries per decidere se una candidate peer mesh STA è raggiungibile oppure no.

- *LongTermTimer*: Timer di lungo termine che schedula l' esecuzione periodica del metodo *handle_lt_expire*; questo si occupa di eseguire una procedura di refresh delle peering instance stabilite con peer mesh STA configurate con una modalità di apertura passiva. Senza questo meccanismo la mesh STA continuerebbe a mantenere i peering stabiliti con le passive-open peer mesh STA che sono state spente e riaccese rapidamente.[2] In tal caso infatti l' handle_st_expire non farebbe in tempo a rilevare lo spegnimento del nodo.

- *LongTermTimeout*: intervallo di tempo con cui Click rischedula l' esecuzione della *handle_lt_expire*.

- *FMLinkTable*: puntatore all' elemento corrispondente alla Routing Table gestita dall' SRCR. Questo elemento viene acceduto per rimuovere i link di livello SRCR, corrispondenti a coppie <LocalIPAddress, PeerIPAddress> relative a nodi che possono comunicare direttamente, qualora questi link non abbiano un peering corrispondente a livello Sme. L' operazione di cancellazione è effettuata per imporre la visione dei peering, offerta dall' Sme, al protocollo di routing.

2 In realtà questo evento è possibile solo se viene ricaricato lo script di click su un router già in esecuzione con la medesima configurazione

- *FMARPTable*: puntatore all' elemento corrispondente alla tabella ARP. Utilizzato dal' Sme per recuperare gli indirizzi IP da quelli MAC o viceversa ogniqualvolta opera sulla struttura dati FMLinkTable.

- *FRRdb*: puntatore all' elemento FRRdb, che viene acceduto dall' Sme per scatenare l' attivazione di un LSP di backup (metodo *activate_fast_reroute*) per raggiungere una mesh STA vicina attualmente irraggiungibile.

- *MetricFromThreshold*: soglia associata al valor medio della *MetricFrom* dei peering, superata la quale l' Sme assume che la peer mesh STA non sia più raggiungibile attraverso un dato peering. Questo evento viene utilizzato dall' Sme per scatenare l' attivazione dell' LSP di backup.

- *MetricToThreshold* soglia associata al valor medio della *MetricTo* dei peering, superata la quale si assume che la peer mesh STA non sia più raggiungibile attraverso un dato peering. Questo evento viene utilizzato dall' Sme per scatenare la attivazione del detour.

- *StartExecution*: marca temporale che registra l' istante in cui viene completata l' esecuzione del metodo *initialize* associato all' Sme; ciò al fine di stimare l' istante di inizio della esecuzione del router stesso.

- *TransitionalInterval*: Nelle prime fasi successive all' installazione degli script sui router i valori delle metriche risentono di un transitorio iniziale che potrebbe scatenare l' attivazione non necessaria di un LSP di backup. Per questo motivo il calcolo del valor medio delle metriche viene ritardato di un intervallo di tempo pari a TransitionalInterval.

- *ScanningInformationTable*: mappa hash che associa ad ogni interfaccia wireless un insieme di marche temporali e di contatori; usate dai metodi *active_scanning* e *passive_scanning* per schedulare la trasmissione periodica dei frame relativi alla fase di discovery sull' interfaccia corrispondente.

- *OutPortTable*: mappa hash che associa a ciascuna delle porte di indice 2 e 3, in uscita dall' *Sme*, l' indirizzo MAC della

interfaccia a cui è connessa. Ciò consente di determinare la porta di uscita su cui trasmettere i messaggi tutte le volte che abbiamo a disposizione il solo indirizzo MAC di destinazione.

4.3 DESCRIZIONE DELLE FUNZIONALITÀ

In merito alle funzionalità offerte, l' Sme implementa:

- Una procedura di candidate peer mesh STA discovery.
- Una MSF consistente con il protocollo di peering management.
- Due meccanismi di peering-failure detection, utilizzati per l' attivazione del fast rerouting.
- Un servizio di blacklisting.
- Un meccanismo per garantire la consistenza delle tabelle CandidatePeer e PeeringInstance.

Prima di proseguire con la descrizione dell' implementazione delle funzionalità è utile dare uno sguardo ai diagrammi di attività (fig. 20, 21) dei due metodi principali dell' Sme, la run_timer e la *push*. In merito alla push, la quale gestisce la ricezione dei frame in ingresso all' Sme, si può notare come il processamento dei frame ricevuti, relativi alla fase di discovery o di peering management, sia possibile solo se l' interfaccia di destinazione è abilitata al servizio mesh. Dopodiché i frame di probe request possono essere processati solo se la mesh STA ha un profilo e una configurazione attivi ed entrambi si riferiscono allo stesso MeshID. Questo perché la mesh STA deve rispondere ai probe request con un frame di probe response, nel quale annuncia il proprio profilo attivo; ma questo può farlo soltanto se già partecipa come candidate peer ad una WMN. Le stesse condizioni sono imposte al processamento dei frame di peering management. Come si evince dal diagramma stesso, nel caso in cui queste condizioni non fossero verificate, i frame in questione vengono scartati.

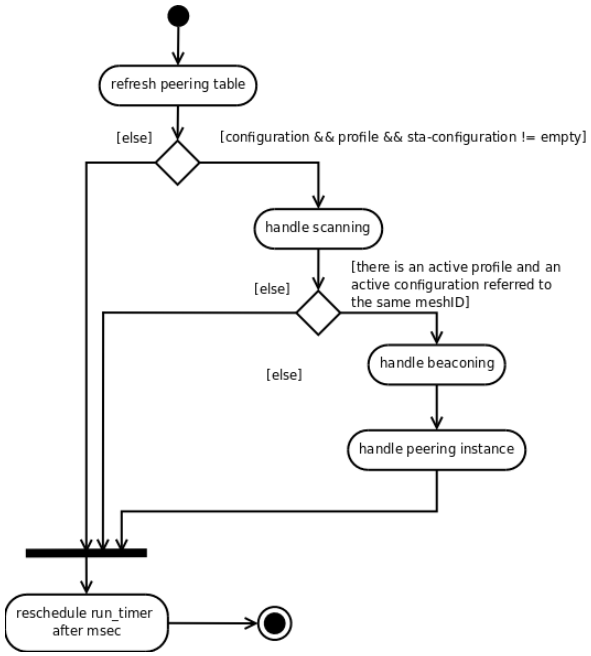

Figura 20: diagramma di attività della run_timer

La run_timer gestisce gli eventi periodici della fase di scanning se l' Sme ha almeno un profilo ed una configurazione, non necessariamente attivi. In più per schedulare la trasmissione periodica dei beacon e per gestire le MSF relative ai peering in fase di apertura/chiusura , l' Sme deve avere un profilo e una configurazione attivi e riferiti allo stesso Mesh ID. L' operazione di *refresh_peering* ha il solo scopo di completare la chiusura dei peering se al momento dell' esecuzione della run_timer l' Sme non ha più un profilo e una configurazione attivi, riferiti allo stesso MeshID.

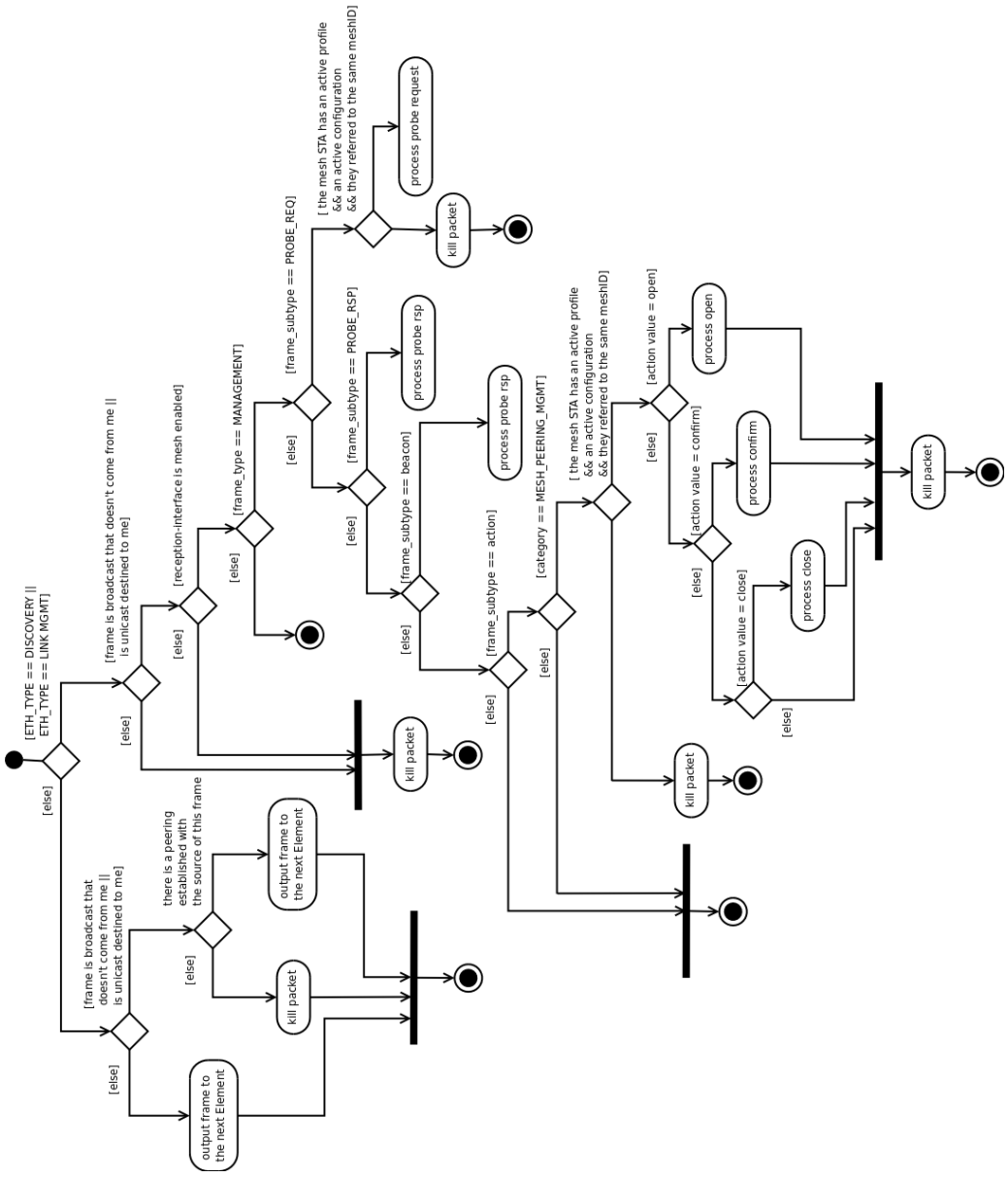

Figura 21: diagramma di attività della push

4.3.1 Candidate peer mesh STA discovery

La procedura di candidate peer mesh STA discovery esegue, per ognuna delle interfacce wireless abilitate al servizio mesh, la operazione di scanning attiva o passiva, in base al valore specificato dal campo scan_mode. Nel primo caso, all' interno della passive_scanning, l' Sme verfica periodicamente se è trascorso un *beacon_interval* senza aver ricevuto un frame di beacon. Se ciò avviene e la mesh STA non ha un profilo attivo, l' Sme attiva il primo profilo disponibile, ma solo in presenza di una configurazione riferita allo stesso MeshID; anche la configurazione viene attivata.

I frame di beacon, ricevuti dalla push, vengono da questa processati all' interno della *process_beacon*. Questa verifica se la mesh STA ha un profilo compatibile con quello annunciato e, nel caso sia presente, lo attiva se ha una configurazione per lo stesso MeshID contenuto nel profilo; aggiunge il profilo annunciato e l' indirizzo MAC dal quale ciò è avvenuto nel vettore CandidatePeer; infine memorizza l' istante in corrispondenza della ricezione del beacon e la modalità di apertura utilizzata dalla candidate peer mesh STA e specificata all' interno dello stesso frame. Queste ultime due informazioni vengono utilizzate per implementare due meccanismi per garantire la consistenza dei vettori PeeringInstance e CandidatePeer, qualora uno o più nodi si spengano e/o riaccendano. Se il processamento del frame va a buon fine viene ricaricato il beacon_interval; ciò ha effetto sulla successiva esecuzione della passive_scanning, che per questo motivo non farà nulla. Il diagramma di attività della passive_scanning è sintetizzato in figura 22.[3]

Nel caso della active_scanning, l' Sme verfica periodicamente se su ciascuna interfaccia wireless locale è stato raggiunto il numero massimo di ritrasmissioni di probe request senza che sia stato ricevuto un probe response. Se ciò avviene, l' Sme attiva il profilo associato al MeshID ricercato solo in presenza di una configurazione per lo stesso MeshID. Se non è stato raggiunto il

3 Non sono stati previsti i diagrammi di attività relativi ai metodi process associati ai vari tipi di frame; questo perchè ad alto livello i diagrammi che li descriverebbero risulterebbero banali

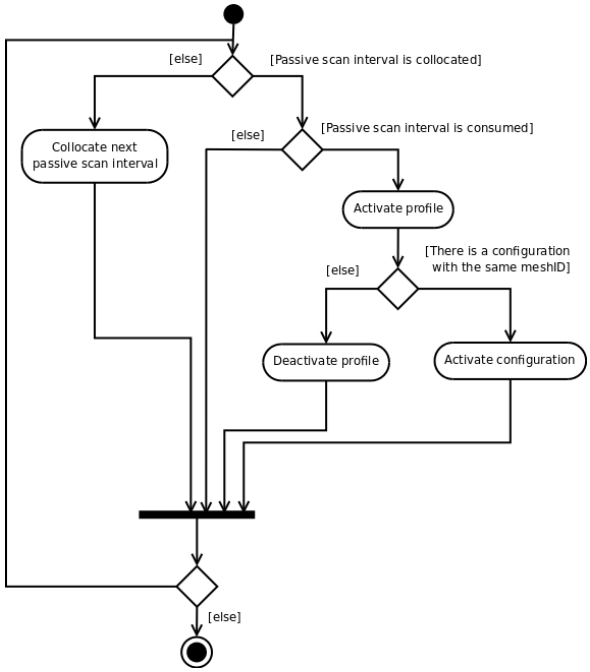

Figura 22: diagramma di attività della passive_scanning

limite al numero di ritrasmissioni, l' Sme calcola un periodo di *active_scan_interval* e attende, tra chiamate successive del metodo stesso, che sia scaduto; quando ciò avviene, aggiunge un intervallo di backoff per diminuire la probabilità di sovrapposizione nella trasmissione dei probe request sui vari nodi; dopodiché trasmette il frame. La trasmissione di un ulteriore probe request viene evitata se è stato ricevuto e correttamente processato un frame di probe response. All' interno della push, il frame viene processato dalla *process_probe_rsp*, la quale esegue le stesse operazioni della process_beacon; infatti nel caso di corretto processamento del frame, verranno attivati profilo e configurazione, se non erano ancora attivi, viene ricaricato il active_scan_interval e, infine, aggiunge l' indirizzo MAC del vicino e il profilo da questo annunciato nel vettore CandidatePeer. In figura 23 viene rappresentato il diagramma di attività della active_scanning. Si osserverà che in seguito i diagrammi di attività si riferiscono alle sole funzioni associate alle attività periodiche.

Dopo che è stato attivato un profilo, la mesh STA inizia a trasmettere i frame di beacon con un periodo nell' intorno di

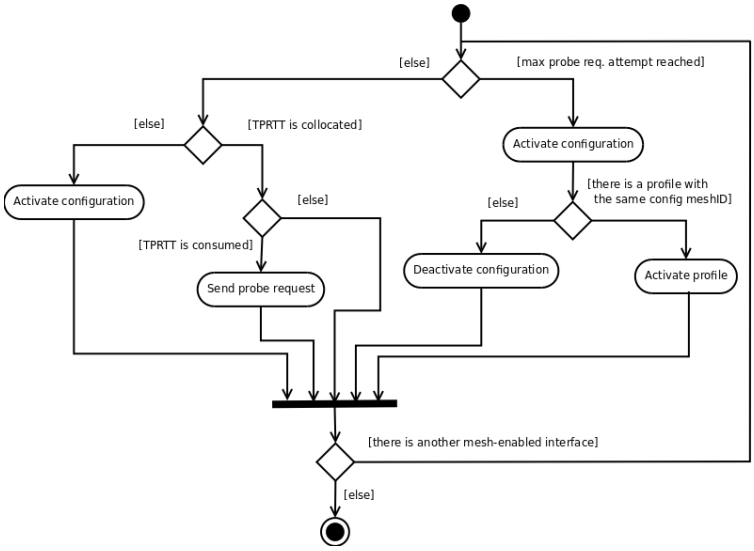

Figura 23: diagramma di attività della active_scanning

beacon_interval. Infatti l' Sme esegue periodicamente la handle_beaconing, attraverso la quale verifica che non sia trascorso più di un beacon_interval dall' ultima trasmissione; quando ciò avviene, aggiunge un intervallo di backoff, scaduto il quale, trasmette il frame di beacon (vedi fig. 24).

4.3.2 Peering management

Se la modalità di apertura impostata è quella attiva, con l' esecuzione periodica della *handle_peering_instance* l' Sme invia un frame di open a ciascuna delle candidate peer mesh STA che non si trovano nella blacklist e verso le quali non ha già aperto una PeeringInstanceEntry. Dopodichè verifica per ciascuna istanza di peering aperta se il relativo timer è scaduto o meno. Se il timer scaduto è quello di retry e non si è raggiunto il numero massimo di ritrasmissioni, viene trasmesso nuovamente un frame di open; se il timer scaduto è quello di confirm, viene avviata la procedura di chiusura attraverso l' invio del frame di close. Se a scadere è il timer di holding, viene comandata la cancellazione della *PeeringInstanceEntry*. Se la mesh STA è impostata con una modalità di apertura passiva, l' Sme creerà una PeeringInstanceEntry dopo aver ricevuto e correttamente processato il frame

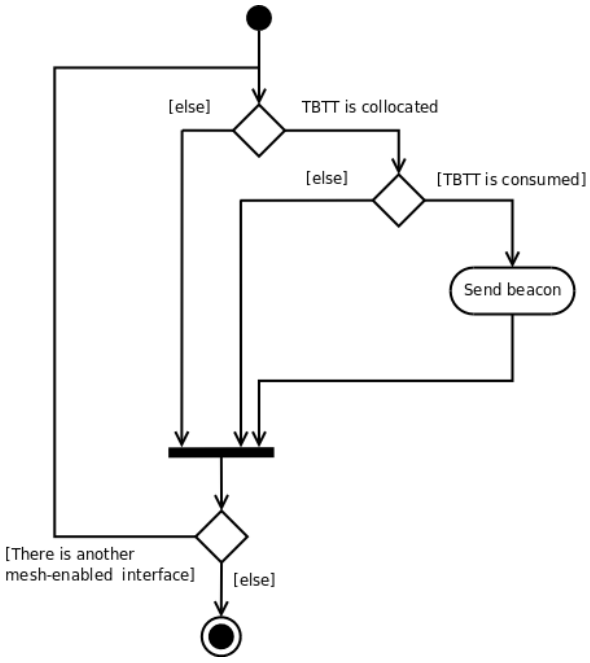

Figura 24: diagramma di attività della handle_beaconing

di apertura all' interno della *process_open*. Coerentemente con le specifche seguite [29], i messaggi di open e di confirm devono specificare il proprio profilo attivo, in modo tale che le due parti possono essere sicure della validità del profilo accettato.

4.3.3 Peering-failure detection e attivazione del fast rerouting

L' *Sme* fornisce due meccanismi per determinare se un peering in stato established o la peer mesh STA corrispondente sia o meno presente. Il primo utilizza la marca temporale LastBeaconReceived, interna alla CandidatePeerEntry, con la quale l' Sme memorizza l' istante di arrivo relativo all' ultimo frame di beacon ricevuto dalla mesh STA e trasmesso da una delle candidate peer mesh STA. L' Sme esegue periodicamente la remove_expired_entries, associata allo ShortTermTimer, controllando da quanto tempo non viene ricevuto un frame di beacon/probe response da ciascuno dei candidate peer conosciuti; se questo intervallo di tempo supera la soglia CpDeadInterval, si assume che il candidate peer non sia più raggiungibile; perciò viene can-

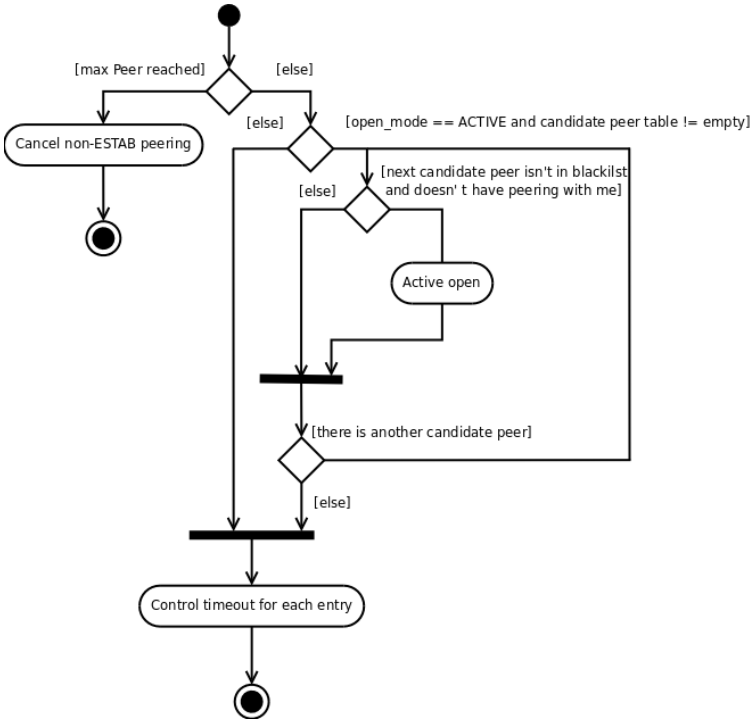

Figura 25: diagramma di attività della handle_peering_instance

cellata l' entrata corrispondente dal vettore CandidatePeer, viene chiuso il peering ad esso associato, se presente, infine, viene attivato l' LSP di backup attraverso l' interfaccia fornita dallo FRRdb.

Il secondo meccanismo si basa sull' utilizzo del valor medio delle metriche associate alle due direzioni del peering. L' Sme recupera i valori istantanei delle metriche dall' elemento *ETTMetric* e li memorizza in due code circolari, l' AverageMetricFromMe e l' AverageMetricToMe, associate a ciascun peering. I valori presenti in queste due code sono poi utilizzati per calcolare una media mobile della metrica in entrambe i versi. Quando il valor medio della metrica in uno dei due versi supera il valore di soglia, l' Sme comanda l' attivazione dell' LSP di backup.

Una differenza sostanziale rispetto al metodo precedente è che in questo caso non viene effettuata la cancellazione né del candidate peer, né dell' eventuale peering stabilito con esso. Infatti il peggioramento del valor medio della metrica agisce su una scala dei tempi molto più grande (uno/due ordini di grandezza)

rispetto a quella su cui si basa il primo meccanismo. Perciò se un nodo continua ad essere rilevato come candidate peer, il fatto che la metrica peggiori può essere messo in relazione con un aumento delle interferenze nella zona limitrofa al nodo oppure con un aumento del carico del traffico.

Un' ultima considerazione che riguarda il calcolo del valor medio delle metriche: nelle prime fasi successive all' avvio della rete i campioni delle metriche hanno dei valori molto alti, dovuti ad una fase di transitorio sperimentata da tutti i nodi. I valori medi risultanti porterebbero ad un' attivazione certa anche se non necessaria del fast rerouting. Per evitare che ciò avvenga il calcolo del valor medio della metrica viene ritardato di un tempo pari al valore del campo TransitionalInterval.

4.3.4 Servizio di blacklist

Questo servizio si basa sull' utilizzo di una coda di indirizzi MAC, alla quale è possibile aggiungere, mediante handler, uno o più indirizzi MAC; essi sono relativi alle interfacce di rete dei candidate peer verso i quali non si intende stabilire né una adiacenza, né un peering. Infatti l' handler, oltre ad aggiungere lo indirizzo MAC alla coda, cancella l' entrata corrispondente nella CandidatePeer e scatena l' azione di chiusura dell' istanza di peering. Per evitare che la successiva ricezione di beacon e di open possa ripopolare le due tabelle con l' indirizzo MAC escluso, i frame vengono scartati dalla push qualora specifichino un indirizzo MAC sorgente presente in blacklist.

4.3.5 Consistenza delle strutture dati

Il meccanismo con cui si determina la presenza/assenza di un nodo, basato sul rilevamento del tempo trascorso dall' ultima ricezione di un beacon dai nodi vicini, fallisce su una mesh STA, se un' altra stazione, impostata con una modalità di apertura passiva, viene riavviata e/o riconfigurata rapidamente, prima

che venga rieseguito il metodo remove_expired_entries.[4] In uno scenario simile, la mesh STA tornerebbe a ricevere i frame di beacon dalle passive-open mesh STA dopo circa un secondo, assumendo quindi che esse siano raggiungibili.[5] In altre parole, la mesh STA conserverebbe tutti i peering stabiliti con le passive-open meshSTA che sono state riavviate rapidamente, le quali, invece, avrebbero la propria PeeringInstance vuota.

Il problema viene risolto facendo in modo che le mesh STA, indipendentemente dalla modalità di apertura adottata, inviino un frame di open ad ognuna delle passive-open mesh STA verso le quali ha già un peering stabilito. Se tale messaggio specifica un localMAC ed un localLinkID, relativi all' identificatore della PeeringInstanceEntry, noti alla candidate-peer meshSTA che riceve il messaggio di open, questa invierà un messaggio di conferma e il link sarà stato "rinfrescato", altrimenti ne viene chiesta la chiusura attraverso l' invio del messaggio di close. L' esecuzione di questa procedura viene scatenata periodicamente dal timer di lungo termine LongTermTimer.

4.4 STRINGA DI CONFIGURAZIONE

Di seguito viene mostrata la dichiarazione dell' elemento Sme presente nello script di configurazione utilizzato nel testbed. Essa specifica la lista completa dei parametri relativi alla stringa di configurazione.

```
sme :: Sme(
      SCAN_MODE "passive",
      OPEN_MODE "active",
      IF_ATH0 00:0B:6B:D9:54:24,
      BEACON_PERIOD_ATH0 350,
      MESH_ENABLED_ATH0 false,
      IF_ATH1 00:0B:6B:D9:92:7B,
```

4 In realtà una situazione del genere si può verificare soltanto rilanciando il file di configurazione di click

5 Questo tempo dipende dal valore di BeaconPeriod specificato nella configurazione dell' Sme. In ogni caso questo deve essere in proporzione più breve rispetto al periodo utilizzato dalla remove_expired_entries per decidere se la candidate peer mesh STA non sia più raggiungibile

```
BEACON_PERIOD_ATH1 350,
MESH_ENABLED_ATH1 true,
ARP mesh_router/arp,
LT mesh_router/lt,
FRRDB mesh_router/frrdb,
MESHID "FluidMesh",
MESH_FORWARD true,
MAX_RETRIES 3,
RETRY_TIMEOUT 1500,
MTTL 31,
CONF_TIMEOUT 1500,
HOLD_TIMEOUT 1500,
ACTIVE_CONFIG true,
PSEL_PROTOID 0,
PSEL_METRICID 0,
CONGESTION_MODE 0,
SYNC_PROTOID 0,
AUTH_PROTOID 0,
ACTIVE_PROFILE false,
METRIC_FROM_THRESHOLD 10000,
METRIC_TO_THRESHOLD 10000,
);
```

```
SCAN_MODE "active" | "passive"
OPEN_MODE "active" | "passive"
```

Questi due parametri prendono come argomenti una tra le due stringhe, `"active"` o `"passive"`, tramite le quali si specifica, rispettivamente, quale modalità di scanning e di apertura deve essere utilizzata dall' Sme. Questi due parametri sono validi su tutte le interfacce wireless della mesh STA abilitate al servizio mesh; sono entrambe opzionali; se non definiti esplicitamente vengono inizializzati implicitamente a `"passive"`.

```
IF_ATHi mac_address
BEACON_PERIOD_ATHi millisec
MESH_ENABLED_ATHi true | false
```

I tre parametri permettono di definire una STAConfigEntry per ciascuna delle interfacce wireless locali. `IF_ATHi` ha come argomento l' indirizzo MAC che identifica l' interfaccia a cui associare tale configurazione. L' indirizzo MAC è espresso nella forma: `XX:XX:XX:XX:XX:XX` dove `XX` sono due cifre esadecimali.

`BEACON_PERIOD` ha come argomento un numero intero che definisce l' intervallo di tempo, in millisecondi, utilizzato come base dall' Sme per calcolare il periodo di trasmissione/attesa dei beacon sull' interfaccia corrispondente. `MESH_ENABLED` ha come argomento un booleano che indica se l' interfaccia wireless in questione è abilitata o meno al servizio mesh.

La configurazione minima richiede che sia specificato il solo indirizzo MAC. I valori di default assegnati agli altri due parametri sono, rispettivamente, `500` e `true`. Ciò significa che ogni indirizzo MAC definito nella stringa di configurazione viene implicitamente abilitato al servizio mesh.

Al momento il numero di configurazioni che si possono fornire è soggetto a due vincoli: si deve definire almeno una configurazione e non se ne possono definire più di due. Questi due vincoli sono stati imposti a seguito dei cambiamenti introdotti nel passaggio al draft D3.0.[6]

```
ARP mesh_router/arp
LT mesh_router/lt
FRRDB mesh_router/frrdb
```

`ARP` ha per argomento il nome dell' elemento FMARPTable, `LT` quello dell' elemento FMLinkTable, mentre `FRRDB` quello associato all' elemento *FRRdb*. Tutti e tre i parametri sono obbligatori perchè necessari a realizzare l' integrazione software tra l' Sme e l' architettura di routing preesistente.

```
MESHID "meshid_name"
```

Parametro opzionale, ha come argomento una stringa che rappresenta l' identificatore della rete mesh. Se definito, ad esso vengono associati un profilo ed una configurazione.

```
MESH_FORWARD true | false
MAX_RETRIES num
RETRY_TIMEOUT millisecs
MTTL num
CONF_TIMEOUT millisecs
HOLD_TIMEOUT millisecs
```

6 Fino al draft D2.06 [28], l' abilitazione del servizio mesh era valida sull' intera mesh STA e non sulla singola interfaccia. Tale cambiamento ha richiesto l' aggiunta della struttura dati STAConfig

```
ACTIVE_CONFIG true | false
```

I parametri elencati sopra sono tutti opzionali e consentono di associare una configurazione al `MESHID` specificato nella stringa di configurazione. Ciascuno di questi parametri è inizializzato con un valore di default, per cui è possibile fornire implicitamente una configurazione. In ogni caso, l' Sme aggiunge la configurazione alla tabella *MeshSTAConfig* solo se è stato definito il parametro `MESHID`.

`MESH_FORWARD` ha come argomento un booleano che indica se la mesh STA è abilitata o meno ad instradare i frame. Valore di default: `true`.[7]

`MAX_RETRIES` prende come argomento un numero intero, che rappresenta il numero massimo di messaggi di open che si possono ritrasmettere nel tentativo di apertura di un peering. Valore di default: `2`.

`RETRY_TIMEOUT` specifica il timeout, espresso in millisecondi, con cui viene impostato il retry timer. Valore di default: `1500`.

`MTTL` ha per argomento un numero intero che rappresenta il numero massimo di hop, all' interno della WMN, che un frame può attraversare prima di essere scartato.

`CONF_TIMEOUT` specifica il timeout, espresso in millisecondi, con cui viene impostato il confirm timer. Valore di default: `1500`.

`HOLD_TIMEOUT` specifica il timeout, espresso in millisecondi, con cui viene impostato l' holding timer. Valore di default: `1500`.

`ACTIVE_CONFIG` prende come argomento un booleano che indica se la configurazione è attiva oppure no. Valore di default: `false`.

```
PSEL_PROTOID 0
PSEL_METRICID 0
CONGESTION_MODE 0
SYNC_PROTOID 0
AUTH_PROTOID 0
ACTIVE_PROFILE false
```

Questi parametri consentono di installare un profilo associato al `MESHID` specificato nella stringa di configurazione. I parame-

7 Attualmente Il campo corrispondente a questo parametro non viene utilizzato, poiché l' Sme può intercettare solo il traffico in ingresso alle interfacce wireless, ma non quello di uscita

tri sono tutti opzionali e se non definiti esplicitamente vengono inizializzati con un valore di default. Come nel caso della configurazione, l' inserimento del profilo all' interno della tabella *MeshProfile* è condizionato dalla presenza o meno del `MESHID` nella stringa di configurazione.

`PSEL_PROTOID` ha per argomento un numero intero che rappresenta l' identificatore del protocollo di routing associato a questo profilo. Valore di default: 0 (nessun protocollo di routing).

`PSEL_METRICID` ha come argomento un numero intero che rappresenta l' identificatore della metrica adottata da questo profilo. Valore di default: 0 (nessuna metrica).

`CONGESTION_MODE` viene inizializzato con un numero intero che rappresenta l' identificatore della modalità di controllo della congestione associata a questo profilo. Valore di default: 0 (nessuna modalità di controllo della congestione specificata).

`SYNC_MODE` viene inizializzato con un numero intero che rappresenta l' identificatore del protocollo di sincronizzazione associato a questo profilo. Valore di default: 0 (nessun protocollo di sincronizzazione specificato).

`ACTIVE_PROFILE` prende come argomento un booleano che indica se il profilo è attivo oppure no.

```
METRIC_FROM_THRESHOLD 10000
METRIC_TO_THRESHOLD 10000
```

`METRIC_FROM_THRESHOLD` prende in ingresso il massimo valor medio relativo alla metrica in uscita dal peering, oltre il quale il peering stesso viene considerato inaccessibile.

`METRIC_TO_THRESHOLD` prende come argomento il massimo valor medio relativo alla metrica in ingresso al peering, oltre il quale il peering stesso viene considerato inaccessibile.

Di seguito viene mostrata una dichiarazione contenente il numero minimo di argomenti per configurare correttamente l' Sme.

```
sme :: Sme(
     IF_ATH0 00:0B:6B:D9:54:24,
     ARP mesh_router/arp,
     LT mesh_router/lt,
     FRRDB mesh_router/frrdb,
     MESHID "FluidMesh");
```

4.5 DESCRIZIONE DEGLI HANDLER

Per accedere allo stato interno dell' Sme sono stati definiti un certo numero di handler, così suddivisi:

- handler di visualizzazione/modifica del contenuto delle tabelle.
- handler di visualizzazione/modifica dei parametri di funzionamento dell' Sme.

Per ciascun handler viene illustrata la sintassi del comando corrispondente e le operazioni da questo svolte.

Handler di lettura

Nome:
`candidate_peer_print_table`
Sinossi:
`#cat candidate_peer_print_table`
Descrizione:
visualizza le entrate della tabella CandidatePeer, ciascuna delle quali specifica l' indirizzo MAC e il profilo annunciato associati ad ognuna delle candidate peer mesh STA.

Nome:
`meshprofile_print_table`
Sinossi:
`#cat mesh_profile_print_table`
Descrizione:
visualizza il contenuto della tabella MeshProfile, costituita dai profili installati sulla mesh STA.

Nome:
`meshstaconfig_print_table`
Sinossi:
`#cat meshstaconfig_print_table`
Descrizione:
visualizza il contenuto della tabella MeshSTAConfig, elencando

le configurazioni definite sulla mesh STA.

Nome:
`staconfig_print_table`
Sinossi:
`#cat staconfig_print_table`
Descrizione:
visualizza il contenuto della tabella STAConfig, costituita dalle configurazioni definite per ciascuna delle interfacce wireless locali.

Nome:
`peer_print_table`
Sinossi:
`#cat peer_print_table`
Descrizione:
visualizza il contenuto della tabella PeeringInstance, mostrando i peering stabiliti o in fase di apertura/chiusura dall' Sme verso le candidate peer o peer mesh STA.

Nome:
`cpeer_blacklist_print_table`
Sinossi:
`#cat cpeer_blacklist_print_table`
Descrizione:
visualizza il contenuto della tabella Blacklist, costituita dagli indirizzi MAC delle candidate peer mesh STA verso le quali si blocca l' apertura dei peering.

Nome:
`scan_mode_print`
Sinossi:
`#cat scan_mode_print`
Descrizione:
visualizza la modalità di scanning attualmente impostata sulla Sme.

Nome:
`open_mode_print`
Sinossi:
`#cat open_mode_print`
Descrizione:
visualizza la modalità di apertura attualmente impostata sulla Sme.

Nome:
`expire_timeout_print`
Sinossi:
`#cat expire_timeout_print`
Descrizione:
visualizza il valore in millisecondi dell' ExpireTimeout.

Nome:
`expire_st_timeout_print`
Sinossi:
`#cat expire_st_timeout_print`
Descrizione:
visualizza il valore in secondi dell' ShortTermTimeout.

Nome:
`expire_lt_timeout_print`
Sinossi:
`#cat expire_lt_timeout_print`
Descrizione:
visualizza il valore in secondi associato a LongTermTimeout.

Nome:
`cpeer_dead_interval_print`
Sinossi:
`#cat cpeer_dead_interval_print`
Descrizione:
visualizza il valore in secondi associato a CpDeadInterval.

Handler di scrittura

Nome:
`meshprofile_add_entry`
Sinossi:
`#echo` *mpentry* `> meshprofile_add_entry`

Descrizione:
aggiunge l' entrata specificata come argomento all' interno della tabella MeshProfile. Se l' entrata aggiunta è attiva, contestualmente viene disattivato, se presente, il profilo al momento attivo.
Esempio di mpentry:
`MyMesh 0 0 0 0 0 true false`
`MyMesh` corrisponde al Mesh ID. I cinque valori a `0` corrispondono agli identificatori associati rispettivamente al PathSelectionProtolID, al PathSelectionMetricID, al CongestionMode, al SynchronizationProtocolID, all' AuthenticationProtocolID; il primo booleano corrisponde all' AcceptPeering, il secondo indica se la entrata è attiva o meno.

Nome:
`meshprofile_remove_entry`
Sinossi:
`#echo` *mp_entry* `> meshprofile_remove_entry`
Descrizione:
rimuove l' entrata specificata come argomento dalla MeshProfile.

Nome:
`meshstaconfig_add_entry`
Sinossi:
`#echo` *msta_entry* `> meshstaconfig_add_entry`
Descrizione:
aggiunge l' entrata specificata come argomento all' interno della tabella MeshSTAConfig. Se l' entrata aggiunta è attiva, contestualmente viene disattivata, se presente, la configurazione al momento attiva.

Esempio di msta_entry:
`MyMesh true 3 1500 31 1500 1500 true`
`MyMesh` corrisponde al Mesh ID. Il primo booleano è associato al campo MeshForwarding. `3` è il valore del campo MeshMaxRetries. `31` corrisponde al valore di MTTL. I valori `1500` corrispondono rispettivamente al RetryTimeout, al ConfirmTimeout e allo HoldingTimeout. L' ultimo valore indica se l' entrata è attiva o meno.
Nome:
`meshstaconfig_remove_entry`
Sinossi:
`#echo` *msta_entry* `> meshstaconfig_remove_entry`
Descrizione:
rimuove l' entrata specificata come argomento dalla MeshSTAConfig

Nome:
`cpeer_blacklist_add_entry`
Sinossi:
`#echo` *mac_address* `> cpeer_blacklist_add_entry`
Descrizione:
aggiunge l' indirizzo MAC specificato alla Blacklist. Contestualmente rimuove il candidate peer, se presente, dalla tabella CandidatePeer, e scatena la chiusura del peering, comanda la cancellazione della corrispondente entrata di livello SRCR (presente come coppia di <IP_from, IP_to>) e contenuta nella FMLinkTable; infine su di essa ricalcola tutte le rotte, eseguendo l' algoritmo di Dijkstra.

Nome:
`cpeer_blacklist_remove_entry`
Sinossi:
`#echo` *mac_address* `>cpeer_blacklist_remove_entry`
Descrizione:
rimuove l' indirizzo MAC specificato come argomento dalla tabella Blacklist.

Nome:
`cpeer_blacklist_clear`
Sinossi:
`#echo > cpeer_blacklist_clear`
Descrizione:
cancella tutte le entrate presenti nella Blacklist.

Nome:
`scan_mode_set`
Sinossi:
`#echo` *mode* `> scan_mode_set`
Descrizione:
imposta la modalità di scanning al valore specificato da *mode*. Valori possibili: `active` o `passive`

Nome:
`open_mode_set`
Sinossi:
`#echo` *mode* `> open_mode_set`
Descrizione:
imposta la modalità di apertura al valore specificato da *mode*. Valori possibili: `active` o `passive`

Nome:
`expire_timeout_set`
Sinossi:
`#echo` *milliseconds* `> expire_timeout_set`
Descrizione:
imposta *ExpireTimeout* il valore specificato da *milliseconds*.

Nome:
`expire_st_timeout_set`
Sinossi:
`#echo` *seconds* `> expire_st_timeout_set`
Descrizione:
assegna a *ShortTermTimeout* il valore specificato da *seconds*.

Nome:
`expire_lt_timeout_set`
Sinossi:
`#echo` *seconds* `> expire_lt_timeout_set`
Descrizione:
assegna a *LongTermTimeout* il valore specificato da *seconds*.

Nome:
`cpeer_dead_interval_set`
Sinossi:
`#echo` *seconds* `> cpeer_dead_interval_set`
Descrizione:
assegna a *CpDeadInterval* il valore specificato da *seconds*.

4.6 INTEGRAZIONE SOFTWARE

Attraverso la procedura di peering management, l' Sme è in grado di stabilire uno o più peering tra le mesh STA vicine a patto che utilizino dei profili tra loro compatibili. L' insieme dei peering stabiliti all' interno della WMN fornisce la vista dei link disponibili ai protocolli di livello superiore; questi dovranno adeguarsi all' infrastruttura di collegamento "soft" fornita dalla Sme. Nondimeno la definizione esplicita dei peering permette all' Sme di gestire meccanismi di fast rerouting, come ad esempio l' attivazione di un detour a seguito del peggioramento della metrica di un peering, e di predisporre meccanismi di traffic engineering, come ad esempio imporre una particolare topologia di rete all' interno di un contesto full-mesh.

Alla luce di queste considerazioni, il processo di integrazione con l' infrastruttura preesistente ha richiesto che l' Sme:

- imponesse la propria visione dei peering all' SRCR e al meccanismo di forwarding.
- recuperasse i valori delle metriche calcolati dall' ETT solo se relativi a link di livello network aventi un corrispondente peering a livello data-link.

- sfruttando l' interfaccia offerta dall' FRRdb, attivasse i detour in seguito alla mancata ricezione, per un dato intervallo di tempo, dei beacon trasmessi da una candidate peer mesh STA, oppure in seguito al superamento di una soglia da parte del valor medio della metrica associato al peering.

L' integrazione con l' SRCR comporta che in tutti i punti in cui l' Sme esegue un' operazione di cancellazione di un peering deve contestualmente effettuare la cancellazione del corrispondente link presente nella *FMLinkTable*. L' operazione di rimozione viene effettuata tramite l' invocazione del metodo *remove_ngh_link(from, to)*; *from* e *to* sono i due indirizzi IP che individuano il link diretto. Una volta cancellato il link bisogna eseguire il metodo *dijkstra(from_me)* sulla tabella di routing, in modo da riportarla ad uno stato consistente; andranno infatti ricalcolate tutte le rotte che utilizzano il link appena rimosso. Lo argomento *from_me* indica se dovranno essere ricalcolate solo le rotte a partire dal nodo in questione o se andranno ricalcolate tutte. Per aggiornare completamente la FMLinkTable il metodo dijkstra andrà quindi rieseguito due volte, con due valori diversi per l' argomento from_me.

L' operazione di rimozione sarebbe inutile se dopo aver completato la cancellazione di un peering permettessimo all' SRCR e a tutti i protocolli di livello superiore di continuare a scambiare pacchetti. In una situazione simile, l' SRCR, continuando a scambiare informazioni di raggiungibilità con i nodi vicini, reinstallerebbe dopo poco tempo il link rimosso e tutte le rotte che lo utilizzano. Questo problema viene risolto facendo in modo che la push filtri tutti i frame, scartando quelli che non riguardano la Sme e che specificano un indirizzo MAC sorgente verso il quale l' Sme stesso non ha stabilito alcun peering.

Per evitare che un link di livello network venga reintrodotto nella FMLinkTable in assenza di un peering corrispondente, bisogna intervenire anche sull' elemento ETTMetric. Questo calcola le metriche associate a ciascun link, misurando e comunicando ai nodi vicini il tasso di perdita relativo al traffico broadcast prodotto dall' elemento *ETTStat*. In particolare, l' ETTMetric aggiorna la metrica attraverso il metodo *update_link()*, che calcola

il valore attuale della metrica e lo assegna al link corrispondente presente nella FMLinkTable. Quest' ultima operazione viene compiuta mediante il metodo *update_ngh_link(from, to, seq, o, metric))*. Questo metodo, operando sulla FMLinkTable, aggiunge il link <*from*, *to*> associato alla metrica nel caso questo non sia presente.

Per evitare ciò, all' interno dell' Sme viene definito il metodo *update_peering_metric*, che viene invocato all' interno della update_link prima che venga scatenato l' aggiornamento della FMLinkTable. Il metodo update_peering_metric dell' Sme può causare la terminazione anticipata del metodo chiamante se per la coppia <from, to>, passata come argomento, non c' è un peering corrispondente.

Quanto spiegato sopra circa l' update_peering_metric, oltre a risolvere il problema visto, costituisce anche il modo in cui la Sme recupera le metriche dall' ETTMetric e le associa ai rispettivi peering.

All' interno dell' Sme, i punti in cui avviene la cancellazione di un link nella FMLinkTable e la riesecuzione di dijkstra sono:

- nell' handler cpeer_blacklist_add_entry
- nel metodo remove_expired_entries, quando si individua una CandidatePeerEntry dalla quale non si riceve un frame di beacon da un certo intervallo di tempo.
- all' interno della update_peering_metric, quando per la coppia di indirizzi passata come argomento non si trova un peering corrispondente.

L' attivazione del detour avviene ogniqualvolta l' Sme considera un peer non più raggiungibile. Per questo il metodo activate_fast_reroute viene invocato all' interno update_peering_metric, se viene rilevato che il valor medio della metrica ha superato quello di soglia, e all' interno della remove_expired_entries dopo che è stato cancellato un link.

L' operazione di cancellazione di un link dalla FMLinkTable operata dall' Sme ha anche un effetto positivo sui tempi di reazione con cui l' LDP ricalcola un LSP a seguito di un' interruzione. L' LDP utilizza le rotte definite dall' SRCR per consegnare

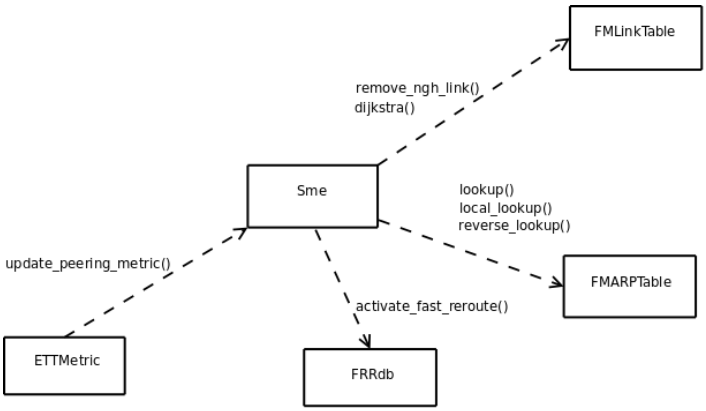

Figura 26: riferimenti definiti per l' integrazione software

i messaggi di label request/reply; ciò significa che gli LSP sono mappati sulle rotte SRCR. Inoltre l' LDP monitora periodicamente la FMLinkTable e in presenza di cambiamenti sulle rotte può decidere di reinstallare gli LSP che le utilizzano. Questo meccanismo viene sollecitato dall' Sme molto più rapidamente di quanto faccia l' SRCR normalemente. Questo ultimo infatti si avvale di un meccanismo a soglia applicato alle metriche per decidere se il link è presente o meno. Tale meccanismo agisce su tempi dell' ordine di qualche minuto. Al contrario l' Sme è in grado di rilevare la assenza di un nodo su tempi dell' ordine di alcuni secondi. In realtà l' integrazione con l' LDP va completata poichè nel caso in cui le topologie di rete siano imposte dalla Sme e non tramite un' opportuna configurazione dei canali, un ingress LSR non riesce ad effettuare l' apertura di un LSP che abbia come egress un nodo verso cui è stata inibita l' apertura del peering.

In figura 26 vengono mostrati i riferimenti tra i vari elementi, definiti per ottenere l' integrazione dell' Sme all' interno dell' infrastruttura di routing preesistente. Ad ogni freccia viene anche associato il metodo invocato attraverso il riferimento.

5

TEST

In questo capitolo verrà trattata la fase dei test. Dopo aver descritto gli scenari adottati (5.1), saranno presentati i test effettuati, accompagnati dai relativi risultati (5.2 e 5.3).

5.1 SCENARI

Nel corso dei test sono stati adottati due scenari differenti: il primo, associato ai test di validazione, prevede che le interfacce wireless di ciascun router siano configurate su due canali (52 e 36) dell' 802.11a, realizzando la topologia magliata di fig. 27. Sono stati collegati due client, rispettivamente al nodo 1 e al nodo 4, per accedere via ssh ai vari dispositivi e controllarne lo stato di funzionamento. La tabella 3 riassume le informazioni associate a ciascun router. Questa tabella specifica, tra le altre cose, le modalità di scanning e di apertura utilizzate nel primo scenario. Si noti che i router 1 e 4 specificano entrambi una modalità di apertura passiva.

Il secondo scenario (fig. 28), associato ai test di prestazione, realizza una topologia semplificata attraverso un' opportuna configurazione dei canali associati alle interfacce wireless. Que-

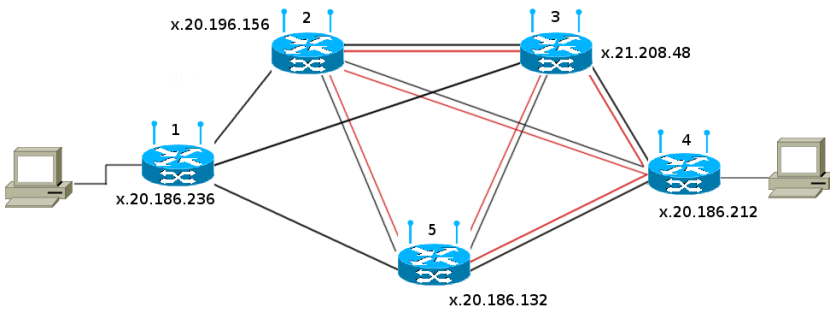

Figura 27: Topologia di rete adottata per i test di validazione

Node num.	Wifi interface	MAC address	Scan mode	Open mode
1	ath0	-	passive	passive
	ath1	00 : 0B: 6B:D9 : 92 : 7B		
2	ath0	00 : 0B: 6B:D9 : 92 : 6B	active	active
	ath1	00 : 0B: 6B:D9 : 92 : 6C		
3	ath0	00 : 0B: 6B:D9 : 92 : 6F	active	active
	ath1	00 : 0B: 6B:D9 : 92 : 17		
4	ath0	00 : 0B: 6B:D9 : 92 : 0A	active	passive
	ath1	00 : 0B: 6B:D9 : 92 : 80		
5	ath0	00 : 0B: 6B:D9 : 92 : 7A	passive	active
	ath1	00 : 0B: 6B:D9 : 92 : 8F		

Tabella 3: Tabella riassuntiva delle informazioni associate a ciascun nodo

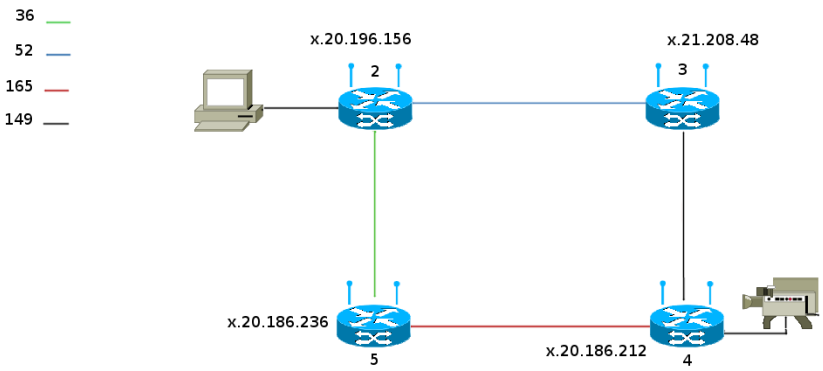

Figura 28: Topologia di rete adottata per i test di prestazione

sta ci garantisce il corretto funzionamento dell' LDP con l' Sme . In questo modo è stato possibile verificare l' efficacia prestazionale del meccanismo di fast rerouting rispetto ai fenomeni di interruzione degli LSP. Lo scenario è composto da quattro nodi, disposti ai vertici di un quadrato, in cui ciascun lato rappresenta il canale a comune tra i due router agli estremi del lato stesso.

I test condotti si concentrano sui casi che abbiamo ritenuto più significativi entro i limiti dell' ambiente di test; a causa delle numero ridotto di nodi a disposizione non è stato possibile eseguire test più complessi.

5.2 TEST DI VALIDAZIONE

I test di collaudo sono stati effettuati seguendo un approccio di tipo black-box, ossia, partendo dalle specifiche del progetto, si è andati a verificare che le funzionalità richieste fossero presenti, cercando contemporaneamente eventuali errori. I test hanno riguardato:

1. La verifica del funzionamento delle funzionalità di gestione della tabelle MeshProfile, MeshSTAConfig.
2. La verifica del funzionamento di apertura/mantenimento/chiusura dei peering tra i nodi predisposti nel testbed, secondo le specifiche previste dal DRAFT 3.0.
3. La verifica delle funzionalità relative alla blacklist.
4. La verifica delle modalità attiva/passiva relative alla fase di apertura dei peering
5. La verfica del meccanismo di rilevamento dell' assenza di una candidate peer mesh STA da parte delle altre.

5.2.1 Test 1

(a) Rimuovere la configurazione attiva, avendo contestualmente un profilo attivo riferito allo stesso Mesh ID. L' obiettivo è quello di dimostrare che le tabelle PeeringInstance e CandidatePeer restano vuote, poiché la mesh STA non ha un profilo e una configurazione attivi.La FMLinkTable dovrà specificare solo link interni al router.

Di seguito viene mostrato il contenuto delle tabelle CandidatePeer, PeeringInstance e MeshSTAConfig associate al nodo 2 e la PeeringInstance associata al nodo 1 prima di rimuovere la configurazione attiva. Per ragioni pratiche viene mostrato solo un sottoinsieme dei campi associati a ciascuna delle entrate delle tabelle.

```
root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID -
     RoutingProtoID - MetricID - CongestionMode - SynchroProtoID -
     AuthProtoID)
```

```
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:6F 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:17 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0

root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState)
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 15014 31367 25089 28948 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 49357 63572 49900 8760 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 49242 48802 34102 5215 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 28389 37838 45089 31126 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 4699 36218 43453 44108 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 43217 15493 54244 20724 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:7B 57067 62798 63325 2996 ESTAB

root@fm-20-196-156:/click/smt# cat meshstaconfig_print_table
Mesh STA Configuration Table: (MeshID - MeshForwarding -
     MeshMaxRetries - MeshRetryTimeout - MeshTTL - MeshConfTimeout -
     meshHoldTimeout - Active)
FluidMesh true 5 1500 31 1500 1500 true

root@fm-20-186-236:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState)
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:8F 65472 30658 8090 16360 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:17 56316 56789 3211 46614 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C 62798 57067 2996 63325 ESTAB
```

Di seguito viene mostrato il contenuto delle stesse tabelle viste sopra dopo che è stata rimossa la configurazione attiva. Come ci si attendeva la CandidatePeer e la PeeringInstance del nodo 2 sono entrambe vuote.

```
root@fm-20-196-156:/click/smt# echo FluidMesh true 5 1500 31 1500
     1500 true>meshstaconfig_remove_entry
root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState -
     metricFromMe - avgMetricFromMe - metricToMe - avgMetricToMe -
     timer type - estabInterval)
no entries found

root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID -
     RoutingProtoID - MetricID - CongestionMode - SynchroProtoID -
     AuthProtoID - Forwarding - AcptPeering - OpenMode - LastRefresh)
no entries found
```

```
root@fm-20-196-156:/click/smt# cat /click/mesh_router/lt/routes
1.20.196.156 5.20.196.156 1 (946698526,49) 1.20.196.156
2.20.196.156 5.20.196.156 1 (946698526,49) 2.20.196.156

root@fm-20-186-236:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState)
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:8F 65472 30658 8090 16360 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:17 56316 56789 3211 46614 ESTAB
```

(b) A partire dallo stato delle tabelle ottenuto dai test precedenti, aggiungere una configurazione attiva, associata ad un Mesh ID differente da quello associato al profilo installato. L' obiettivo è dimostrare che le tabelle considerate rimangano vuote.

```
 root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID -
     RoutingProtoID - MetricID - CongestionMode - SynchroProtoID -
     AuthProtoID - Forwarding - AcptPeering - OpenMode - LastRefresh)
no entries found
root@fm-20-196-156:/click/smt# echo fluidmesh true 5 1500 31 1500
     1500 true>meshstaconfig_add_entry
root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID -
     RoutingProtoID - MetricID - CongestionMode - SynchroProtoID -
     AuthProtoID - Forwarding - AcptPeering - OpenMode - LastRefresh)
no entries found

root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState -
     metricFromMe - avgMetricFromMe - metricToMe - avgMetricToMe -
     timer type - estabInterval)
no entries found
```

5.2.2 Test 2

Con i test precedenti e con quelli effettuati al punto 3 la verifica della corretta gestione della PeeringInstance è ottenuta in maniera implicita.

5.2.3 Test 3

Aggiungere/rimuovere l' indirizzo di un peer alla/dalla Blacklist. L' obiettivo è verificare che localmente la PeeringInstance

e la CandidatePeer vengano aggiornate coernetemente. Le tabelle FMLinkTable sui vari nodi devono riflettere questa situazione. Di seguito viene mostrato il contenuto delle tabelle PeeringInstance, CandidatePeer e Blacklist relative al nodo 2, prima che venga aggiunta nella Blacklist l' indirizzo MAC dell' interfaccia di rete wireless operante sul nodo 1. La rotta associata alla destinazione 1 è stata contrassegnata con un asterisco.

```
root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 1708 42175 28558 33597 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 19741 36828 32918 35608 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 50585 8074 40112 9174 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 54986 43614 36296 1847 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 35279 48696 32561 44424 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 34048 21962 17440 45600 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:7B 19254 20888 44546 21109 ESTAB

root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID - RoutingProtoID
     - MetricID - CongestionMode - SynchroProtoID - AuthProtoID)
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:6F 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:17 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0

root@fm-20-196-156:/click/smt# cat /click/mesh_router/lt/routes
1.20.196.156 5.20.196.156 1 (946687617,64) 1.20.196.156
2.20.196.156 5.20.196.156 1 (946687617,64) 2.20.196.156
3.20.186.132 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946699598,1)
     3.20.186.132
3.20.186.212 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946687680,1)
     3.20.186.212
3.20.196.156 5.20.196.156 1 (946687617,64) 3.20.196.156
3.21.208.48 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946687679,1)
     3.21.208.48
4.20.186.132 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946699598,1)
     3.20.186.132 1 (946699521,79) 5.20.186.132 1 (946699521,79)
     4.20.186.132
4.20.186.212 5.20.196.156 1 (946687617,64) 4.20.196.156 400 (946687679,2)
     4.20.186.212
4.20.186.236 5.20.196.156 1 (946687617,64) 4.20.196.156 400 (946687679,2)
     4.20.186.236
4.20.196.156 5.20.196.156 1 (946687617,64) 4.20.196.156
4.21.208.48 5.20.196.156 1 (946687617,64) 4.20.196.156 400 (946687680,0)
     4.21.208.48
```

```
5.20.186.132 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946699598,1)
     3.20.186.132 1 (946699521,79) 5.20.186.132
5.20.186.212 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946687680,1)
     3.20.186.212 1 (946699521,79) 5.20.186.212
*5.20.186.236 5.20.196.156 1 (946687617,64) 4.20.196.156 400 (946687679,2)
     4.20.186.236 1 (946687635,44) 5.20.186.236
5.21.208.48 5.20.196.156 1 (946687617,64) 3.20.196.156 400 (946687679,1)
     3.21.208.48 1 (946699521,79) 5.21.208.48

root@fm-20-196-156:/click/smt# cat cpeer_blacklist_print_table
BlackList Table: (EtherAddress)
no entries found
```

Dopo aver aggiunto l' indirizzo MAC `00:0B:6B:D9:92:7B` alla blacklist del nodo 2, si può notare come nelle tabelle PeeringInstance e CandidatePeer non ci sia più alcun riferimento a tale indirizzo. La FMLinkTable non specifica alcuna rotta per raggiungere il nodo 1.

```
root@fm-20-196-156:/click/smt# echo 00:0B:6B:D9:92:7B>cpeer_blacklist_add_
     entry

root@fm-20-196-156:/click/smt# cat cpeer_blacklist_print_table
BlackList Table: (EtherAddress)
00:0B:6B:D9:92:7B

root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 1708 42175 28558 33597 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 19741 36828 32918 35608 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 50585 8074 40112 9174 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 54986 43614 36296 1847 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 35279 48696 32561 44424 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 34048 21962 17440 45600 ESTAB

root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID - RoutingProtoID
     - MetricID - CongestionMode - SynchroProtoID - AuthProtoID)
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:6F 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0
00:0B:6B:D9:92:17 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0

root@fm-20-196-156:/click/smt# cat /click/mesh_router/lt/routes
1.20.196.156 5.20.196.156 1 (946687737,102) 1.20.196.156
2.20.196.156 5.20.196.156 1 (946687737,102) 2.20.196.156
```

```
3.20.186.132 5.20.196.156 1 (946687737,102) 3.20.196.156 428 (946699755,3)
     3.20.186.132
3.20.186.212 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687836,3)
     3.20.186.212
3.20.196.156 5.20.196.156 1 (946687737,102) 3.20.196.156
3.21.208.48 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687833,3)
     3.21.208.48
4.20.186.132 5.20.196.156 1 (946687737,102) 3.20.196.156 428 (946699755,3)
     3.20.186.132 1 (946687721,117) 5.20.186.132 1 (946687721,117)
     4.20.186.132
4.20.186.212 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687836,3)
     3.20.186.212 1 (946687721,117) 5.20.186.212 1 (946687721,117)
     4.20.186.212
4.21.208.48 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687833,3)
     3.21.208.48 1 (946687721,117) 5.21.208.48 1 (946687721,117) 4.21.208.48
5.20.186.132 5.20.196.156 1 (946687737,102) 3.20.196.156 428 (946699755,3)
     3.20.186.132 1 (946687721,117) 5.20.186.132
5.20.186.212 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687836,3)
     3.20.186.212 1 (946687721,117) 5.20.186.212
5.21.208.48 5.20.196.156 1 (946687737,102) 3.20.196.156 400 (946687833,3)
     3.21.208.48 1 (946687721,117) 5.21.208.48
```

5.2.4 Test 4

Impostare la modalità di apertura passiva sulla coppia di nodi 1 e 2. L' obiettivo è verificare che le tabelle PeeringInstance siano vuote. Inoltre Le tabelle FMLinkTable sui due nodi devono riflettere questa situazione.

Di seguito viene mostrato il contenuto di CandidatePeer e PeeringInstance prima che sul nodo 2 venga rieseguito il file di configurazione in cui si specifica una modalità di apertura passiva. La ultima entrata della tabella CandidatePeer sul nodo 2 evidenzia come il nodo 1 sia già configurato con una modalità di apertura passiva. Di seguito viene mostrata la tabella PeeringInstance del nodo 1 prima e dopo la reinstallazione del file di configurazione sul nodo 2.

```
 root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID - RoutingProtoID
     - MetricID - CongestionMode - SynchroProtoID - AuthProtoID OpenMode)
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 A
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 P
00:0B:6B:D9:92:6F 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 A
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 P
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 A
00:0B:6B:D9:92:17 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 A
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 P
```

```
root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 1708 42175 28558 33597 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 19741 36828 32918 35608 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 50585 8074 40112 9174 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 54986 43614 36296 1847 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 35279 48696 32561 44424 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 34048 21962 17440 45600 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:7B 54697 39888 58124 11315 ESTAB (*)

root@fm-20-186-236:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:8F 65472 30658 8090 16360 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:17 56316 56789 3211 46614 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C 39888 54697 11315 58124 ESTAB (*)
```

Dopo aver ricaricato il file di configurazione con una modalità di apertura passiva, la tabella PeeringInstance viene ripopolata con i soli peering relativi alle active-open peer mesh STA.

```
root@fm-20-196-156:/click/smt#cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 52536 2539 9122 29121 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 29122 23679 46977 11305 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 29669 9997 40915 46985 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 7029 10037 42409 37308 ESTAB

PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:8F 65472 30658 8090 16360 ESTAB
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:17 56316 56789 3211 46614 ESTAB
```

5.2.5 Test 5

Spegnere il nodo 3. L' obiettivo è mostrare come il contenuto delle tabelle CandidatePeer e PeeringInstance del nodo 2 venga correttamente aggiornato a seguito di questo evento. Nel listato qui sotto sono indicate con un asterisco le entrate relative al nodo 3.

```
root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:17 52211 61342 18527 59278 ESTAB (*)
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:6F 30811 9596 49050 6592 ESTAB (*)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 19522 54640 33068 47039 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 28857 12579 8732 10308 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:7B 6340 63725 35822 46903 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 3920 57823 23708 0 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 27022 19278 16976 0 ESTAB

root@fm-20-196-156:/click/smt# cat candidatepeer_print_table
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID - RoutingProtoID
     - MetricID - CongestionMode - SynchroProtoID - AuthProtoID - Forwarding
      - AcptPeering - OpenMode - LastRefresh)
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true A
00:0B:6B:D9:92:17 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true A (*)
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 true true A
00:0B:6B:D9:92:6F 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 true true A (*)
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true P
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 true true P
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true P
```

A seguito dello spegnimento del nodo 3, le tabelle CandidatePeer e PeeringInstance del nodo 2 vengono correttamente aggiornate. Si noti infatti l' assenza sia della CandidatePeerEntry che della PeeringInstanceEntry relative al nodo 3.

```
CandidatePeer Table: (CandidatePeerMAC - LocalMAC - MeshID - RoutingProtoID
     - MetricID - CongestionMode - SynchroProtoID - AuthProtoID - Forwarding
      - AcptPeering - OpenMode - LastRefresh)
00:0B:6B:D9:92:8F 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true A
00:0B:6B:D9:92:7A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 true true A
00:0B:6B:D9:92:80 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true P
00:0B:6B:D9:92:0A 00:0B:6B:D9:92:6B FluidMesh 0 0 0 0 0 true true P
00:0B:6B:D9:92:7B 00:0B:6B:D9:92:6C FluidMesh 0 0 0 0 0 true true P

root@fm-20-196-156:/click/smt# cat peer_print_table
PeerInstance Table: (localMAC - peerMAC - localLinkID - peerLinkID -
     localAssociationID - peerAssociationID - peeringState - metricFromMe -
     avgMetricFromMe - metricToMe - avgMetricToMe - timer type -
     estabInterval)
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:80 19522 54640 33068 47039 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:0A 28857 12579 8732 10308 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:7B 6340 63725 35822 46903 ESTAB
00:0B:6B:D9:92:6B 00:0B:6B:D9:92:7A 3920 57823 23708 0 ESTAB
00:0B:6B:D9:92:6C 00:0B:6B:D9:92:8F 27022 19278 16976 0 ESTAB
```

5.3 TEST DI PRESTAZIONE

Per verificare l' efficacia del modulo di link management nel garantire un reinstradamento veloce a seguito del guasto/spegnimento di un nodo è stato eseguito un test di trasmissione video con due configurazioni differenti dei router:

- con SRCR
- con SRCR, MPLS, LDP, Sme

In entrambi i casi è stato interrotto il percorso di rete utilizzato per instradare il flusso video spegnendo il nodo intermedio. Abbiamo misurato i tempi di reazione delle due configurazioni relativi alla durata dell' interruzione del flusso video.[1]

Sono state eseguite diverse prove e in relazione alla prima configurazione è stata riscontrata una durata dell' interruzione video intorno ai due minuti. Questo ritardo è dovuto al tempo necessario all' SRCR per aggiornare le FMLinkTable sui vari nodi. Con la seconda configurazione, invece, la durata dell' interruzione è stata in un intorno ai 5-10 secondi. Questo dipende dal valore specificato dal campo CpDeadInterval. Nei test questo intervallo è stato impostato a 6 secondi.

1 Per motivi di tempo non è stato possibile effettare una campagna di test per ottenere dei dati statisticamente affidabili

6

CONCLUSIONI

Questo lavoro di tesi si è incentrato sulla realizzazione di un modulo di link management per WMN, conforme alle specifiche del draft 3.0 dell' 802.11s, e sulla sua integrazione all' interno di un testbed di sviluppo, costituito da una rete wireless mesh 802.11 implementata tramite software Roofnet basato su Click, con estensioni per l' inoltro dei pacchetti tramite label switching MPLS, un protocollo di distribuzione delle label ispirato a LDP e un meccanismo di fast-rerouting per la protezione degli LSP tramite dirottamento temporaneo del traffico su percorsi alternativi (detour). All' inizio sono state apportate le modifiche allo script di configurazione per poter utilizzare l' Sme all' interno del driver di Click. Successivamente siamo passati alla realizzazione dell' Sme, implementando la procedura di mesh STA discovery e di peering management, a cui si è aggiunta la definizione di una Blacklist utile per imporre particolari topologie di rete a partire da un contesto mesh. In seguito è stata definita una prima procedura di attivazione dei detour, basata su un meccanismo a soglia associato al valor medio delle metriche dei peering. I campioni delle metriche sono quelli relativi alla metrica ETT, calcolata dall' elemento Click ETTMetric e da questo fornita all' Sme. La difficoltà di scegliere soglie valide e la lentezza intrinseca di un meccanismo di attivazione dei detour basato sulle metriche sono state le due motivazioni che hanno spinto a realizzare un ulteriore meccanismo che riscontrasse in tempi più brevi l' irraggiungibilità di un nodo, permettendo una più rapida attivazione dei detour e contestualmente un più rapido aggiornamento della tabella di routing; il nuovo criterio si è basato sul rilevamento del tempo trascorso dall' ultima ricezione di un beacon per decidere se una mesh STA vicina sia disponibile o meno. Nella fase finale è stata completata l' integrazione della Sme con l' architettura di routing preesistente garantendo che la tabella di routing si adeguasse alla visione topologica del-

la rete fornita di volta in volta dall' Sme. Al termine del lavoro sono stati condotti una serie di test di validazione che hanno verificato il corretto funzionamento dei servizi offerti dall' Sme e la corretta integrazione di quest' ultima con il protocollo di routing esistente (SRCR). Una serie di test di prestazione ha dimostrato l' efficacia del meccanismo di rilevamento dell' assenza di un nodo nell' abbattere i tempi di convergenza della tabella di routing; inoltre, utilizzando topologie ottenute con un' opportuna configurazione dei canali, è stata dimostrata l' efficacia del fast rerouting nel garantire continuità di servizio a seguito dello spegnimento/guasto di un nodo intermedio appartenente allo LSP utilizzato. Ulteriori test andrebbero effettuati per poter individuare soglie relative al valor medio delle metriche che siano efficaci nel garantire continuità di servizio; l' Sme andrebbe inoltre dotato di un proprio meccanismo di misura della qualità dei peering che adotti una metrica alternativa a quella ETT. Durante la validazione, è stata rilevata la mancanza di convergenza del modulo LDP alle nuove rotte calcolate dal processo di routing in alcun scenari particolari, la cui analisi e soluzione sono state demandate ad una fase successiva a questo lavoro.

BIBLIOGRAFIA

[1] M. Atzori, S. Rossi, Thesis: *Realizzazione e analisi delle prestazioni di un prototipo di rete wireless mesh 802.11 con funzionalità di label switching*. 2008 (Citato alle pagine 3, 9, 12 e 30.)

[2] R. Bruno, M. Conti, E. Gregori, *Mesh Networks: Commodity Multihop Ad Hoc Networks*. Marzo 2005 (Citato alle pagine 7 e 14.)

[3] 802.15.5 TG homepage `http://www.ieee802.org/15/pub/TG5.html` (Citato a pagina 14.)

[4] 802.11s TG report page `http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm` (Citato a pagina 14.)

[5] 802.16a TG homepage `http://ieee802.org/16/tga/index.html` (Citato a pagina 14.)

[6] 802.20 TG homepage `http://grouper.ieee.org/groups/802/20/` (Citato a pagina 14.)

[7] G.R. Hiertz, S. Max, R. Zhao, D. Denteneer, L. Berlemann, *Principles of IEEE 802.11s*. 2007

[8] X. Wang, A.O. Lim, *IEEE 802.11s wireless mesh networks: Framework and challenges*. Ottobre 2007

[9] IEEE 802.11 Standard Working Group, *Draft Standard for Information Technology - Telecommunications and Information Exchange Between Systems - LAN/MAN Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE P802.11-REVma/D9.0*. Gennaio 2007 (Citato a pagina 45.)

[10] J. Bicket, D. Aguayo, S. Biswas, R. Morris, *Architecture and Evaluation of an Unplanned 802.11b Mesh Network*. Settembre 2005 (Citato alle pagine 10, 11, 18, 19, 25 e 28.)

[11] D. Aguayo, J. Bicket, R. Morris, *SrcRR: A High Throughput Routing Protocol for 802.11 Mesh Networks*. Maggio 2006 (Citato a pagina 11.)

[12] J. Bicket, D. Aguayo, S. Biswas, G. Judd, R. Morris, *A measurement study of rooftop 802.11b mesh network*. In Proc. ACM SIGCOMM Conference (SIGCOMM 2004). Settembre 2004

[13] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris *A high-throughput path metric for multi-hop wireless routing*. In Proc. of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03), San Diego, California. Settembre 2003 (Citato a pagina 28.)

[14] R. Draves, J. Padhye, B. Zill *Comparison of Routing Metric for Static Multi-Hop Wireless Networks*. 2004

[15] K. Xu, M. Gerla, S. Bae, *Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad-hoc networks*. Ad Hoc Network Journal. Luglio 2003 (Citato a pagina 19.)

[16] H. Lundegren, K. Ramachandran, E. Belding-Royer, K. Almeroth, M. Benny, A. Hewatt, A. Touma, A. Jardosh, *Experiences from the Design, Deployment and Usage of the UCSB MeshNet TestBed*. Aprile 2006 (Citato a pagina 11.)

[17] A. Zimmermann, M. Güneç, M. Wenig, U. Meis, J. Ritzerfeld, *How to study Wireless Mesh Networks: A hybrid Testbed Approach*. 2007 (Citato a pagina 11.)

[18] Multiprotocol Label Switching `http://www.ietf.org/rfc/rfc3031.txt`. Gennaio 2001 (Citato a pagina 29.)

[19] E. Kholer, Master Thesis: *Click Modular Router*. 2001 (Citato alle pagine 20 e 21.)

[20] E. Kholer, R. Morris, B. Chen, J. Jannotti, M.F. Kaashoek *The Click Modular Router, ACM Transactions on Computer Systems*. Agosto 2000 (Citato a pagina 21.)

[21] *Click Modular Router*. http://read.cs.ucla.edu/click/ (Citato alle pagine 21 e 24.)

[22] *Click Modular Router Element Class Reference*. http://www.read.cs.ucla.edu/click/doxygen/ (Citato alle pagine 24 e 50.)

[23] *PATS: Performance Analysis of Telecommunication Systems*. http://www.pats.ua.ac.be/software/click (Citato a pagina 24.)

[24] *OpenWrt homepage* http://openwrt.org/ (Citato a pagina 20.)

[25] *Linux Wireless*. http://linuxwireless.org/en/developers/Documentation/mac80211

[26] *WifiMesh - FreeBSD*. http://wiki.freebsd.org/WifiMesh

[27] IEEE 802.11s Task Group, *Draft Standard for Information Technology - Telecommunications and informations exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment undefined: Mesh Networking, Unapproved draft P802.11s/D1.08*. Gennaio 2008

[28] IEEE 802.11s Task Group, *Draft Standard for Information Technology - Telecommunications and informations exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 10: Mesh Networking, Unapproved draft P802.11s/D2.06*. Gennaio 2009 (Citato a pagina 72.)

[29] IEEE 802.11s Task Group, *Draft Standard for Information Technology - Telecommunications and informations exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 10: Mesh Networking, Approved draft P802.11s/D3.0*. Marzo 2009 (Citato alle pagine 4, 36, 43, 47, 49, 51 e 67.)

[30] F. Giurlanda, Thesis: *Sviluppo e validazione di un protocollo di scambio delle label con supporto al fast reroute in reti wireless-mesh basate su MPLS*. Giugno 2009 (Citato alle pagine 3, 20, 32, 34 e 36.)